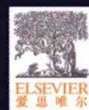


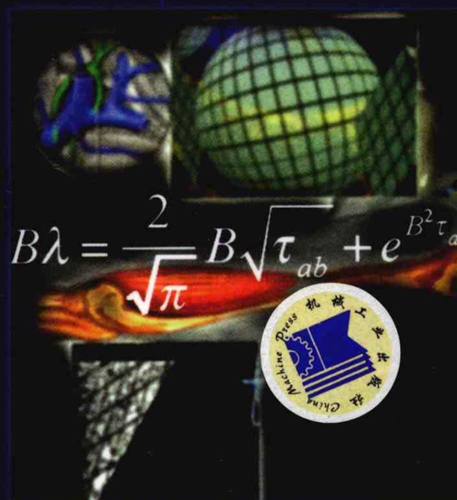
国际信息工程先进技术译丛



数值方法在生物医学工程中的应用

Numerical Methods in Biomedical Engineering

Stanley M.Dunn
（美） Alkis Constantinides 著
Prabhas V.Moghe
封洲燕 译



机械工业出版社
CHINA MACHINE PRESS





数值方法在生物医学工程中的应用



国际信息工程先进技术译丛

数值方法在生物医学工程中的应用

Numerical Methods in Biomedical Engineering

（美）Stanley M. Dunn，Alkis Constantinides，
Prabhas V. Moghe　著
封洲燕　译



机 械 工 业 出 版 社

本书是一部全面介绍生物医学领域仿真建模和数值求解方法的教科书，主要包括4部分内容：生物医学系统的特性和行为，以及数学建模的基础知识；稳态系统建模的方法和常规求解方法，并举例说明了这些方法在分子、细胞和生理各个不同层次的生物医学系统中的应用；动态生物系统的建模和数值求解方法，重点分析了含有常微分方程或偏微分方程的多维生物医学系统的动态模型；举例说明数值方法在复杂生物系统仿真建模中的综合应用。本书的内容十分广泛和全面，所介绍的数值方法不仅包括代数方程、常微分方程和偏微分方程的求解方法，还包括数值解的稳定性分析和统计分析等内容，以及MATLAB、Simulink编程仿真工具和PHYSBE等仿真系统。

书中列举了大量生物医学研究领域的问题求解实例，所涉及的范围非常广泛，并且都给出了完整的仿真计算程序，实用性很强。因此，本书不仅可以作为国内生物医学工程专业本科生或研究生课程的教材，还可以作为生物学、生物材料、生物化学、生物物理学以及生理学等其他专业开设的定量生物学课程的教材，另外，对于致力于生物医学系统仿真建模和数值分析的其他技术人员和研究人员也是一本很好的参考书。

Translation from the English language edition:
Numerical Methods in Biomedical Engineering ISBN: 9780121860318
By Stanley M. Dunn, Alkis Constantinides, Prabhas V. Moghe

本书版权登记号：图字01-2008-2080

图书在版编目（CIP）数据

数值方法在生物医学工程中的应用/(美）史丹利（Stanley，M. D.），(美）阿尔齐斯（Alkis，C.），(美）帕拉巴斯(Prabhas，V. M.）著；封洲燕译. —北京：机械工业出版社，2009. 1
（国际信息工程先进技术译丛）
ISBN 978-7-111-25365-5

Ⅰ. 数… Ⅱ. ①史…②阿…③帕…④封… Ⅲ. 数值计算—应用—生物医学工程
Ⅳ. R318

中国版本图书馆CIP数据核字（2008）第161751号

机械工业出版社（北京市百万庄大街22号 邮政编码100037）
策划编辑：牛新国 责任编辑：牛新国 朱 林 版式设计：霍永明
责任校对：李 婷 封面设计：马精明 责任印制：李 妍
北京富生印刷厂印刷
2009年1月第1版第1次印刷
169mm×239mm · 33.25印张 · 647千字
0001—3000册
标准书号：ISBN 978-7-111-25365-5
定价：78.00元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
销售服务热线电话：(010) 68326294
购书热线电话：(010) 88379639 88379641 88379643
编辑热线电话：(010) 88379178
封面无防伪标均为盗版

译 者 序

近年来生物医学工程学科发展迅速，国内许多大学都开设了生物医学工程专业。但是，正如本书作者在前言中所言，针对该专业本科生教学的教材却寥寥无几，特别是有关数值计算和生物医学系统仿真建模的教材，本书是第一本。作为工程专业之一，数值计算是生物医学工程本科生教学的重要内容；但是，目前国内各大学在开设该专业数值计算课程时，普遍采用工科类的通用教材，内容与生物医学领域所涉及的数值问题缺乏紧密的联系。因此，本书的翻译、出版将给生物医学工程专业的本科教学提供一本合适的数值计算教材。

本书由美国 Rutgers 大学的 3 位教授合作完成，2006 年由 Elsevier 出版公司出版，英文版是该校生物医学工程专业的本科教材。本书的特色是：深入浅出并系统全面地介绍了数值计算的基本理论和各种方法，以及 MATLAB、Simulink 等计算工具的使用方法；大量列举了分子、细胞和生理等各个不同层次生物医学系统的建模和求解实例，所涉及的研究内容非常广泛，包括生物力学、组织工程、信号处理和图像处理、分子生物学、生物医学仪器、药物代谢动力学等，很好地反映了生物医学工程交叉学科的特点；无论是对于数值计算的基本方法，还是对于生物医学系统的建模实例，书中都提供了完整的 MATLAB 程序，详细说明了各种数值解法；各章最后附有练习题，适合作为教材使用。另外，书中所有程序均可以从网站下载（http：//books. elsevier. com/companions/0121860310）

在翻译过程中，对原书某些明显的笔误或印刷错误已作了更正，为保持译著的简洁流畅，没有加以标注。

本书的翻译得到了国家自然科学基金项目的资助（项目编号为 30570585 和 30770548）。在本书完稿之际，特别感谢我丈夫徐政的长期支持和鼓励，以及对部分译稿的仔细校读和修正。另外，郑晓静和徐琼璟同学参与了部分工作，在此深表谢意。

限于译者水平，书中难免存在错误和不妥之处，恳请广大读者批评指正。译者联系方法：hnfzy@ yahoo. com. cn。

封洲燕

2008 年 5 月

浙江大学生物医学工程与仪器科学学院

生物医学工程教育部重点实验室

前　　言

生物医学工程学科正在快速发展，本教科书的目的是全面介绍该领域数值计算问题的求解工具。虽然生物医学工程已经成为广泛普及的工程专业之一，但是，针对这个专业本科生教学的教科书却寥寥无几，而关于数值方法在生物医学工程中应用的教科书还没有。因此，编写本书的目的之一就是要填补这个空白。

本书计划用作生物医学工程专业本科生课程的主要教材。作者已经在 Rutgers 大学生物医学工程系三年级秋季学期开设的数值方法课程中使用了本教材，该课程也可以在三年级或四年级的其他学期开设。如果降低生物医学方面的要求并且调整微积分方面的内容，该书也可以作为二年级课程的教材。修读本书之前，学生需要预修的课程有微积分（Ⅰ～Ⅳ）、一年级的化学和物理学、普通生物学以及生物医学工程导论。使用本书作为教材的数值方法课程可以与三年级的其他课程，如生物医学物质运输、生物医学热力学和动力学、生物力学以及生物医学仪器学等同时开设。本书讲述了各种生物医学系统，很适合于培训对于生物医学工程各个主要研究领域感兴趣的人员。本书还可以作为细胞生物学、生物材料以及生物化学等生命科学专业开设的定量生物学课程的教材。

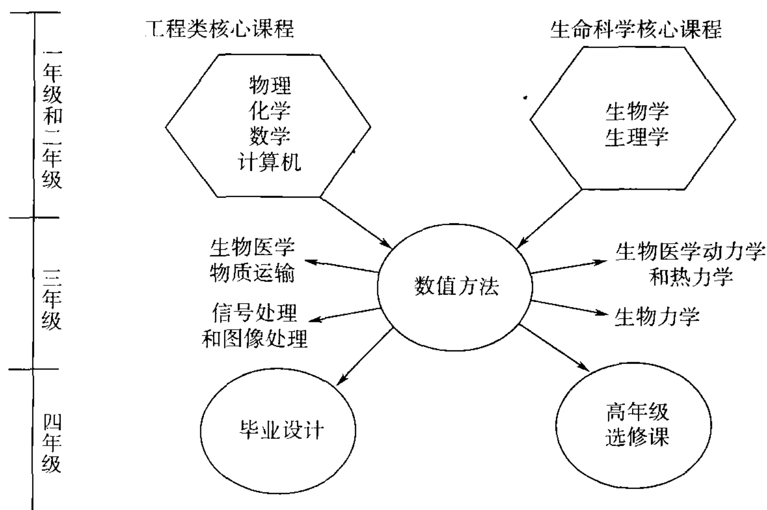


图 1　数值方法是工程理论与生物医学应用之间相互沟通的桥梁

图 1 表示了我们所理解的数值计算方法在培养现代生物医学工程人员中所起的作用。此课程安排在三年级前期，作为多门课程的会聚点，汇集了基础知识和生物医学应用问题的求解技能。课程满足两个主要目标：一是吸收消化学生们已经学习的计算机工具知识，二是将这些工具应用于现代生物医学工程中遇到的各种数学模型。

生物医学工程面临的主要挑战是这个领域所涉及的问题非常广，其范围几乎是无限的。从分子水平到肉眼可见的宏观层面，形成了许多不同的工程分支，例如：化学工程师处理的分子和组织工程方面的问题；机械工程师可以解决的生物力学方面的问题；电子工程师处理的图像学和测量方面的问题等。我们希望把这些通常分开的各种技术统一起来，全面地归纳和总结生物医学领域所遇到的主要问题。表 1 列出了几类需要应用数值方法求解的问题。

表 1　利用本书讲述的数值方法和建模技术可以有效处理的各种不同系统的生物医学问题

	生理系统	细胞和组织系统	分子系统
非线性系统：根的求解	血压 灌流的摩擦系数	细胞迁移 细胞消融	受体与配体的结合 酶反应动力学
数值线性代数	生物医学图像重建	细胞迁移 肌肉骨骼生物力学	酸碱平衡 物质代谢平衡
积分和微分	心律失常检测 心律变化	细胞迁移 细胞生长过程中的摄氧率	药物代谢动力学分析 熔解曲线分析
常微分方程	膜肺氧合 血液氧合	毛细管物质运输 细胞生物力学 细胞膜与动作电位	代谢途径 酶促反应
偏微分方程	阻抗图像 毛细管物质输送	细胞在生物材料上的迁移 激光消融	药物输送 代谢途径
建模工具	腹膜透析 诱发电位	酶促反应参数 肝素转化 细胞迁移	配体结合

本书的安排和概要

表 2 列出了本书的安排，全书共分 4 个部分，外加附录，内容十分广泛全面。

第 1 部分：基础知识

介绍生理系统的特性和行为，以及如何应用数学模型的方法建模仿真生理系统的行为。这部分还讲述各种不同的模型及其相关的数值求解方法，并且讲解如何应用计算机语言编程求解这些问题。

第 2 部分：系统的稳态行为

分析各种系统的线性和非线性代数模型，导出求解模型所需的数值方法。并举例说明数值方法在分子、细胞和生理各个不同层次系统中的应用。讲述用于线性系统的高斯消元法、高斯-若尔当消元法和高斯-赛德尔迭代法，以及用于非线性方程组的线性插值、牛顿-拉弗森法、牛顿法等经典数值方法及其应用。

第 3 部分：系统的动态行为

重点分析含有常微分方程或偏微分方程的多维系统动态模型。用有限差分法推导积分算法和微分算法。讲解用于积分求积和牛顿-柯特斯公式、用于常微分方程的欧拉法和龙格-库塔法及其应用。并且，分析用于偏微分方程求解的几种有限差分法的稳定性。

第 4 部分：建模工具及其应用

结合前面各部分介绍的多种方法，求解复杂生物医学系统的数学模型。

附录

附录 A 是 MATLAB 入门教程。希望同学们阅读这部分内容，复习 MATLAB 语言及其功能。附录 B 主要介绍 MATLAB 的 Simulink 工具箱。附录 C 概述线性代数基础知识，第 4 章和第 6 章所讲述的数值方法将用到这部分知识。附录 D 介绍常微分方程和偏微分方程的解析求解方法，有助于学生和教师利用这些方法验证第 7 章和第 8 章所讲述的数值方法的正确性。附录 E 总结数值稳定性的有关问题。

表 2 《数值方法在生物医学工程中的应用》概况

第 1 部分：基础知识		
第 1 章 生物医学系统建模 生物医学工程 生物医学工程的基本特点 创建工程模型 利用计算机求解生物医学模型的范例 本书概况	**第 2 章 计算技术入门** 引言 计算机在生物医学工程中的角色 程序设计语言工具及方法 MATLAB 的数据结构基础 面向对象系统简介 算法分析和程序分析	**第 3 章 数值分析的概念** 科学计算 数值算法及其误差 泰勒级数展开以及误差 MATLAB 的浮点数表达

（续）

<table>
<tr><td colspan="3">第 2 部分：系统的稳态行为</td></tr>
<tr><td colspan="2">第 4 章　生物医学系统中的线性模型
线性生物医学系统举例
线性代数方程组：
高斯消元法，高斯-若尔当消元法，迭代法
线性系统求解的迭代法：
雅可比法和高斯-赛德尔法
应用举例：
生物力学中的力平衡
生物医学图像及图像处理
代谢工程和细胞生物技术</td><td>第 5 章　生物医学系统中的非线性模型
非线性方程的一般形式
非线性生物医学工程问题举例
逐次代换法
试位法（线性插值）
牛顿-拉弗森法
牛顿法求解非线性方程组
应用举例：
导液管的摩擦系数
Michaelis-Menten 动力学方程
心室血压的测量
受体-配体动力学</td></tr>
<tr><td colspan="3">第 3 部分：系统的动态行为</td></tr>
<tr><td>第 6 章　有限差分法、插值法和积分法
符号算子
向前、向后和中心有限差分法
等距节点插值：
格雷戈里-牛顿法，斯特林插值
插值多项式
非等距节点插值：
拉格朗日法和样条
积分公式：
牛顿-柯特斯积分公式</td><td>第 7 章　动态系统：常微分方程
常微分方程分类
标准型的转化
非线性常微分方程：
欧拉法和龙格-库塔法
线性常微分方程：
特征值方法
稳态解和稳定性分析
数值稳定性
应用举例：
药物代谢动力学：药物吸收问题
酶促反应
活细胞的糖酵解途径
细胞膜动力学和神经细胞电位
干细胞分化动力学
组织工程：细胞迁移</td><td>第 8 章　动态系统：偏微分方程
多维分布系统
偏微分方程分类
初始条件和边界条件
偏微分方程求解：
椭圆型、抛物型、双曲型极坐标
稳定性分析
应用举例：
耳蜗基底膜的动力学
人体白细胞的迁移
药物扩散
激光消融</td></tr>
</table>

（续）

第4部分：建模工具及其应用	
第9章 测量、建模与统计分析 数值方法的作用 测量与误差 描述性统计学和推断统计学 最小二乘法模型、曲线拟合 傅里叶变换 应用举例： MRI图像和CT图像亮度的统计计算 DNA芯片分析中的假设检验 质谱数据分析 EEG信号频率成分分析	**第10章 生物医学系统的建模仿真：应用举例** 生物医学系统的数学模型 PhysioNet、PhysioBank和PhysioToolkit 信号处理 应用举例： ECG仿真，EEG仿真，葡萄糖调节模型，糖尿病及其胰岛素治疗，肾清除率，刚体运动 PHYSBE仿真系统 常规PHYSBE仿真，主动脉缩窄、主动脉瓣狭窄、室间隔缺损、左心室肥大、以及心室压力-容积环的Simulink模型

附录		
附录A：MATLAB简介 附录B：Simulink简介	附录C：线性代数及其相关MATLAB指令概述	附录D：微分方程的解析解 附录E：数值稳定性等问题

书中各章都给出了可行的示例，说明各种数值方法的使用。绝大部分示例都需要运行计算机程序求解，程序用MATLAB语言编写，适用于MATLAB 7.0及以上版本。多数示例中，算法的实现都尽可能编写成通用的MATLAB函数，以便用于其他同类问题的求解。

本书中出现的所有MATLAB程序可以在本书网站（http://books.elsevier.com/companions/0121860310）下载。建议读者访问该网站，以便获取更新信息、勘误表以及作者联系方式等。

致谢：作者感谢惠特克基金（Whitaker Foundation），并特别感谢John Linehan博士支持、审查并批准此课程申报该基金的教材项目。项目的时间规划很有用，使作者在繁忙的两年时间里完成了这项工作。同时感谢Rutgers大学的同事们帮忙选定目前生物医学工程中面临的数值问题作为本书的范例，他们是：John Li、Charlie Roth、David Shreiber和Martin Yarmush。生物医学工程方面新近出版的优秀著作也给作者提供了广泛的论题，有关内容都在书中有所引用。Rutgers大学生物医学工程系2003年和2004年的三年级学生们，特别是Devin Fensterheim、Ronn Friedlander、Kelly Horn、Jessica Nikitchuk、Nayyereh Rajaei、Jonathon Scarpa、Ashley Winter和Jillian Zaveloff等同学，为第10章的编写作出了重要的贡献，并阅读了本书初稿，非常感谢他们提供宝贵的反馈意见。

Stanley M. Dunn
Alkis Constantinides
Prabhas V. Moghe

目　录

第2部分 系统的稳态行为

第3部分　系统的动态行为

第4部分　建模工具及其应用

附 录

第1部分 基础知识

第1章 生物医学系统建模

1.1 生物医学工程

生物医学工程是工程技术的一个分支，用来解决生物学和医学中的有关问题。本书是面向生物医学领域工程人员的一本数值方法入门教程。数值方法就是利用计算机进行精确、有效、稳定的计算的一种数学方法，它是工程人员利用计算机分析系统行为的工具。本书讲述的数值方法用于求解生物医学系统的数学模型。

生物医学领域的工程人员利用电子、机械、化学、材料以及计算机工程等许多传统学科的原理、方法和技术解决各种生物医学难题。所涉及的技术包括：电子工程方面的电路和系统、图像和图像处理、仪器、测量和传感器等；机械工程方面的流体力学、固体力学、热传导、机器人和自动化以及热力学等；化学工程方面的物质运输现象、聚合物和材料、生物技术、药物设计以及药品生产等。

工程人员如果要应用这些理论解决生命科学和医疗保健领域中的各种问题，就必须同时了解和熟悉系统水平、细胞水平以及分子水平的生物学解剖知识和生理知识。如果涉及医疗保健领域的工作，还需要熟悉心血管系统、神经系统、呼吸系统、循环系统、肾脏系统以及体液系统等生理知识。

其他一些名称，如生物工程、临床工程、组织工程等用于表示生物医学工程的不同分支。其中，生物工程处理生物学问题以及生物系统和生理系统之间的关系；临床工程处理医疗保健系统以及病人监护等临床问题；组织工程这个研究方向是应用工程技术设计开发人造组织和装置，用于替代缺失的生物组织结构及其受损功能。利用组织工程的方法，可以将细胞、工程材料以及合适的生物化学因子结合起来，形成一种复合材料，用于改善或替代生物功能，提高医疗效果。

虽然，生物医学工程对于临床医学最突出的贡献是开发仪器设备，用于诊断、治疗和康复。但是，本书收集了来自生物和医学各个领域的示例，读者通过这些示例可以看到生物医学工程能够解决和正在解决的问题所涉及的范围非常

广泛。

生物医学工程的研究领域正在迅速扩大，例如，从生物纳米技术到辅助设备、从分子和细胞工程到手术机器人、从神经肌肉系统到人工肺等等，该学科的工作人员将在生命科学研究以及医疗系统仪器设备的开发中发挥主要作用。本书讲述的内容将为生物医学工程人员进入各个研究领域的工作打下基础。

有关生物医学工程的发展历史有好几种版本，对于该学科的创建时间以及重要的里程碑事件都各有说法。生物医学工程的开创时间可以追溯到 17 世纪、18 世纪或者 19 世纪，这与生物医学工程的定义有关。读者可以参考 Nebeker（2002）的论文，该文章全面介绍了该领域的工作人员为了解决临床医疗和生命科学中所产生的问题，制造各种用于诊断和治疗的仪器设备的历史过程。

1.2 生物医学工程的基本特点

任何生物医学工程装置都要完成一项或多项测量、模拟或者操作性的任务。测量就是检测所研究的物理系统、化学系统或者生物系统的某些特征。作为从事生物医学工程的人员，很重要的一点就是要认识到，测量值是不会完全准确的，因此，需要使用本书后面讲到的统计学方法。还要了解测量的可变性和噪声来源，了解测量仪器的精度、分辨率和重复性等各种测量指标。

生物医学工程领域的操作性任务是指以某种方式与某个系统进行交互作用。生物医学工程主要通过诊断仪器或治疗设备与人体或者生物系统进行交互作用。仪器设备的开发首先要根据需求和限制条件制定技术指标，然后进行设计、制作和测试。

工程建模是指用数学公式描述所研究的现象或系统的物理、化学和生物原理。数学模型是系统及其环境之间相互作用的精确表述。利用模型这个工具，研究人员可以预测系统参数的变化过程。

工程模型就是数学公式，建立公式要用到代数、微积分、微分方程和统计学这 4 种连续数学中的一种或数种。但是，除非只是纯粹的建模，否则，任何主要的生物医学工程模型的求解、信号的处理以及系统的控制都要用计算机系统来实现。因此，必须将连续数学模型的代数方程、积分和微分方程或者变量表达式转化成计算机可以处理的形式，这里的难题是要最大程度地保证准确度和分辨率，并且保证计算过程的稳定性，这就是数值方法需要解决的问题。

1.3 创建工程模型

工程应用需要解决的问题都基于被研究对象与其周围环境之间的物理关系。

由于生物系统中出现的现象范围非常广，因此，重要的是要有一种共同语言用于描述生物电、生物力学和生物化学等各种现象。

1.3.1　问题求解的基本步骤

问题求解的关键是要确定被研究现象所遵守的守恒定律。生物医学工程问题的求解可以分成以下 4 个步骤：

1. 确定需要分析的系统

系统可以是空间的一个区域，也可以是一组待分析的量，它是总体的一个部分。环境就是系统之外的所有部分。系统边界是系统与环境之间无限薄的分隔界面，可以实际存在，也可以是虚构的。边界没有质量，只是作为系统范围的界限。

2. 确定需要考虑的广度性质

广度性质在一个点上是没有值的，其值取决于系统范围的大小，例如，系统的质量正比于系统的大小。系统广度性质的值可以由组成该系统的各个子系统的广度性质的值加和求得。系统广度性质的值只随时间变化。常用的广度性质有质量和体积等。

如果科学证明某个广度性质既不能创造也不能消灭，那么，这个广度性质称为守恒性质。

根据文献报道的实验数据，电荷、线性动量和角动量是守恒的；质量和能量则在下列限制条件下守恒：

1）如果有运动，系统的运动速度要远小于光速；

2）与量子力学的时间尺度相比，时间要很长；

3）不存在核反应。

生物医学系统不太可能违背条件 1）和条件 2），其事件发生的最短时间尺度是 10^{-8} 数量级，比核反应事件的时间还是要长。但是，放射性摄像和核医疗等系统会产生核反应，要谨慎处理这些系统中的质量守恒和能量守恒。

在建立数学模型时，质量、电荷、能量以及动量的守恒都很有用。除了守恒性质以外，其他还有一些广度性质具有有限的生成量和消耗量。典型的有热力学第二定律及其性质——熵。用守恒方程表示时，容易看出熵只能在系统内部产生，并且，对于系统内部的可逆过程，熵的产生率减为 0。

3. 确定要分析的时间段

系统的整个变化经历称为过程。工程分析的目标通常是预测系统的行为，也就是系统经历一个特殊过程时，其中的各个状态的变化轨迹。根据所涉及的时间段不同，过程可以分成 3 种：稳态过程、有限时间过程和暂态过程。

4. 写出守恒定律的数学公式

系统广度性质的值可以随时间变化，并且，这种变化只能通过以下两种机制发生：

1）广度性质通过系统边界的传输；

2）广度性质在系统内部的产生和消耗。

因此，系统内部广度性质的变化与通过系统边界的传输量以及系统内的产生和消耗量有关。这种广度性质的计算原理就是所大家熟悉的平衡方程。虽然平衡方程可以用于系统的任何广度性质，但是，它对守恒性质特别有用。平衡方程有两种形式，即累计形式和速率形式。

1.3.2 建立守恒定律的数学公式

1. 累计形式

累计形式用于分析有限长度的时间段，因此，该形式适用于建立稳态和有限时间过程的模型方程。对于系统的输入输出净传输量，只要计算给定时间段内进入系统的总量，再减去同一时间段内排出系统的总量就可以了。累计形式的计算式为

〈系统内净累计量〉=〈系统净传输量〉+〈系统净生成量〉

对于守恒性质 P，则有

$$P_{系统终值} - P_{系统初值} = (P_i - P_o) + (P_G - P_C)$$

其中，等式左边为系统内净累计量；等式右边第一项为系统净传输量，即输入量减去输出量；等式右边第二项为系统净生成量，即产生量减去消耗量。

守恒定律累计形式的优点是数学方程为代数方程或者积分方程，其缺点是进出系统的量并不是一定能确定的。

2. 速率形式

平衡方程的速率形式与累计形式相似，只是时间段的长度取无限小，相应的变化量的极限值就成了变化速率。将累计形式的数学公式除以时间间隔 Δt，并使 $\Delta t \to 0$，就可以得到速率形式的数学公式，即

〈系统 t 时刻变化率〉=〈系统 t 时刻传输率〉+〈系统 t 时刻生产率〉

对于守恒性质 P，则有

$$\frac{\mathrm{d}P}{\mathrm{d}t} = (\dot{P}_i - \dot{P}_o) + (\dot{P}_G - \dot{P}_C)$$

其中，等式右边第一项为通过系统边界的输入与输出传输率之差；等式右边第二项为系统产生率与消耗率之差。这两项的总和为系统性质随时间的变化率。速率形式的优点是根据物理定理通常很容易测定某个量的变化率，其缺点是数学公式为微分方程。

虽然平衡方程可以用于任何广度性质，但是，当系统的传输项和系统的生成项（即产生/消耗）具有物理意义时，它尤其有用。如果事先已知系统生成项的有关情况，则该方程最有用。

例如，对于守恒的广度性质，其生成项等于0，平衡方程就很简单，其累计形式为

$$P_{系统终值} - P_{系统初值} = P_i - P_o$$

1.3.3 平衡方程的应用

生物医学工程问题的数学方程都是基于一个或者多个化学量和物理量的守恒定律。只要方程建立正确，求解就不难了。以上两小节分别介绍了建模和问题求解的基本步骤，以及守恒定律的累计形式和速率形式。累计形式用于稳态和有限时间问题的求解，而速率形式则用于暂态问题的求解。下面举例介绍应用不同形式守恒定律建立问题求解方程的方法。

例 1.1 如何从守恒定律推导 Nernst 方程

请说明如何利用 1.3.1 节介绍的守恒定律导出 Fick 定律、欧姆定律以及爱因斯坦方程，并说明如何利用这 3 个守恒模型导出 Nernst 方程。本例题取自 Enderle 等人（2000）著作的 3.4 节。

解：

物质守恒定律的速率形式为

$$\frac{\mathrm{d}m_{sys}}{\mathrm{d}t} = \sum_{inlets} \dot{m}_i - \sum_{outlets} \dot{m}_e$$

该方程的含义是，系统总的质量变化率等于系统的输入质量变化率总和与输出质量变化率总和之差。

Nernst 方程考虑的系统是细胞膜，膜外为细胞外液，离子可以跨细胞膜流动。假设 K^+、Na^+、Cl^- 中只有某种离子可以跨膜流动，并且，从胞外向胞内流动的方向设为正向。则根据以上守恒定律的速率形式，可以导出以下 3 个关系式：

1）Fick 定律描述在细胞膜跨膜离子浓度梯度作用下离子的扩散流动，它表示化学势对于系统物质守恒的作用。守恒定律等式左边的物质流动速率就是进入系统的物质流量，由扩散引起。等式右边为进出系统的物质变化速率的总和，也就是通过细胞膜离子通道的流量。根据物质守恒定律的速率形式，可得

$$J_{扩散} = -D\frac{\mathrm{d}I}{\mathrm{d}x}$$

即离子扩散流量等于离子的跨膜浓度梯度乘以扩散系数 D，也就是离子流量是离子浓度梯度作用的结果。式中的负号表示离子沿着浓度降低的方向扩散。

2）欧姆定律描述在细胞膜跨膜电动势作用下离子的流动。它也是基于物质守恒定律的，但是，等式的右边为其他带电粒子形成的电场作用下所产生的物质流量。电场 $\vec{E}$ 的作用形成电位变化率 $\mathrm{d}v/\mathrm{d}x$，该电位变化产生了等式右边的离子流量，即

$$J_{迁移} = -\mu Z[I]\frac{\mathrm{d}v}{\mathrm{d}x}$$

式中　$J_{迁移}$——电场引起的离子流量；

μ——迁移率；

Z——离子价数；

$[I]$——离子浓度；

v——电位。

3）爱因斯坦方程是动量守恒定律的一种形式，描述扩散与离子迁移之间的关系，说明电场作用产生的离子流动与浓度差的渗透压作用产生的离子流动互相平衡，由守恒定律

$$\frac{\mathrm{d}P_{sys}}{\mathrm{d}t} = \sum F_{ext} + \sum_{inlets}\dot{P}_i - \sum_{outlets}\dot{P}_o$$

可得

$$D = \frac{kT\mu}{q}$$

式中　k——玻耳兹曼常数 1.38×10^{-23}J/K；

T——单位为开尔文（Kelvin）的绝对温度；

q——电子所带电荷量 1.61×10^{-19}C。

Fick 定律和欧姆定律都表示物质守恒，根据工程上的叠加原理，将两式相加，就得到离子总流量，即

$$J = J_{扩散} + J_{迁移} = -D\frac{\mathrm{d}[K^+]}{\mathrm{d}x} - \mu Z[K^+]\frac{\mathrm{d}v}{\mathrm{d}x}$$

稳态时，离子总流量为0。并将爱因斯坦方程代入，则有

$$0 = J_{扩散} + J_{迁移} = -\frac{kT}{q}\mu\frac{\mathrm{d}[K^+]}{\mathrm{d}x} - \mu Z[K^+]\frac{\mathrm{d}v}{\mathrm{d}x}$$

对于钾离子 K^+，有 $Z=1$。并求跨膜电位的积分

$$\int_{v_o}^{v_i}\mathrm{d}v = -\frac{kT}{q}\int_{K_o^+}^{K_i^+}\frac{\mathrm{d}[K^+]}{[K^+]}$$

可得

$$v_i - v_o = -\frac{kT}{q}\ln\frac{[K^+]_i}{[K^+]_o}$$

上式就是 Nernst 方程，表示细胞膜跨膜电位差是驱动离子输运的化学势的函数。

1.4　利用计算机求解生物医学模型的范例

上一节讲述了求解生物医学问题和设计诊断治疗仪器的基本方法。但是，只有问题的解析解是不够的，仪器设备的设计和控制都要用到计算机，也就是说，我们必须懂得如何将连续数学模型转化为离散数学（即数值分析）模型，然后编写计算机程序完成方程的计算（即数值方法）。本节将举例说明如何在生物医学工程中应用数值分析和数值方法。

1.4.1　实时 PCR 效率的建模仿真

PCR 是多聚酶链式反应（Polymerase Chain Reaction）的简称，该技术可以将 DNA 短片段（通常少于 3000 对碱基）扩增百万倍，用于测定 DNA 的大小以及核苷酸序列。

需要扩增的 DNA 片段称为目标序列，由一对特殊的 DNA 引物识别，引物的长度通常约为 20 个脱氧核苷酸。脱氧核苷酸共有 4 种，为了方便，将它们简称为 dNTP。

PCR 反应分成 3 个主要步骤，这些步骤要重复 30 次或 40 次（Hsu 等人，1996）。反应过程由 PCR 仪自动循环完成，PCR 仪可以在很短的时间内加热和冷却试管中的混合反应物。如图 1.1 所示，PCR 反应的 3 个步骤如下：

1）94℃变性：变性期 DNA 双链解开，成为单链，并且所有酶促反应停止。

2）54℃退火：单链引物与单链模板 DNA 之间不断配对结合和解离，完全配对的结合比较稳定，可以保持较长时间不解离，聚合酶就可以连接到这种模板和引物形成的 DNA 双链上，准备复制模板。如果有几个 dNTP 加入之后，模板与引物之间结合的离子键就很强了，不会再解离。

3）72℃延伸：随着 dNTP 的加入，引物就与模板牢固结合。与模板互补的 dNTP 连接在引物的 3' 端，不断延伸。温度升高后，没有完全配对的引物又产生松动，这些片段就不会延伸。

PCR 反应过程中两条 DNA 单链都被复制，因此，基因序列的复制数目以指数形式增长。

实时 PCR 技术简称 rtPCR，通常用于监测基因表达的进程及其序列信息。在 rtPCR 中，通常假设扩增效率为恒定的常数。但是，数据分析表明扩增效率并不是恒定不变的，而是循环次数的函数（Gevertz 等人，2005）。基于这个现象，通过了解扩增效率的行为特征，可以改进 rtPCR 的定量分析方法。rtPCR 的数学模型可以揭示扩增效率的变化规律，提供通过 rtPCR 数据定量分析基因表达水平的

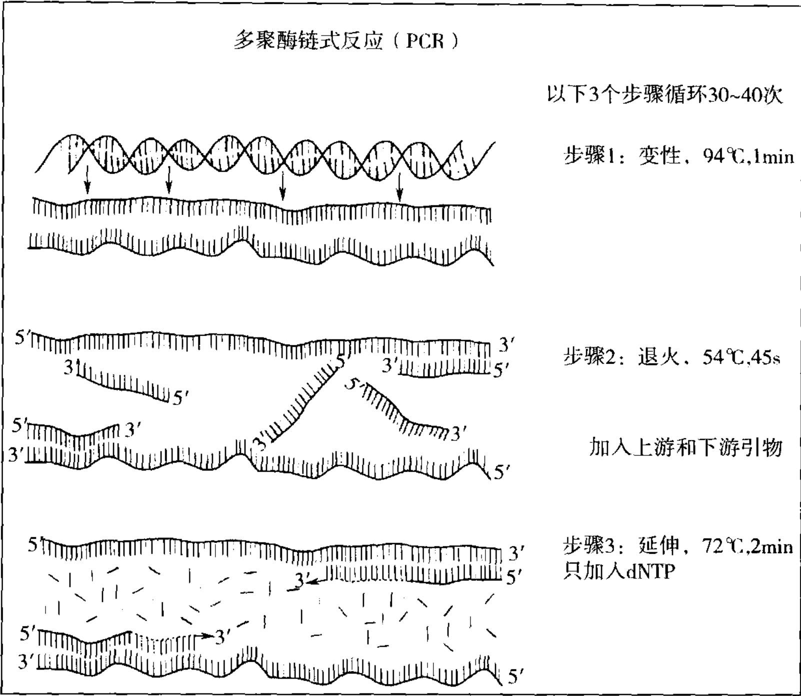


图 1.1 PCR 反应中基因数目以指数形式扩增［摘自 Vierstraete（1999）］

更准确的方法。

退火和延伸的数学模型如下：DNA 双链变性之后，解链形成两条 DNA 单链，分别称为模板 1（T1）和模板 2（T2）。当温度降到 54℃时，退火阶段就开始了。引物 P1 应该与 T1 退火，形成杂交混合双链 H1；引物 P2 则应该与 T2 退火，形成杂交混合双链 H2。这两个反应的化学方程式为

$$P1 + T1 \leftrightarrow H1 \tag{1.1}$$

$$P2 + T2 \leftrightarrow H2 \tag{1.2}$$

然而，在 PCR 的退火阶段也会发生其他反应。两条模板链 T1 和 T2 是碱基互补的，相遇时会重新结合，形成杂交混合模板双链 HT，反应式为

$$T1 + T2 \leftrightarrow HT \tag{1.3}$$

最后，与引物的设计有关，还可能形成引物二聚体，反应式为

$$P1 + P2 \leftrightarrow D \tag{1.4}$$

由此可见，退火阶段会同时发生 4 种反应，这些反应可以用热力学方程（稳态）模拟，也可以用动力学方程（暂态）模拟。

稳态热力学方程模型可以跟踪 PCR 退火阶段各个产物的总浓度。将物质平衡原理应用于式（1.1）~式（1.4），可得以下 4 个方程：

$$
\begin{aligned}
[P1]_T &= [P1] + [H1] + [D] \\
[P2]_T &= [P2] + [H2] + [D] \\
[T1]_T &= [T1] + [H1] + [HT] \\
[T2]_T &= [T2] + [H2] + [HT]
\end{aligned} \tag{1.5}
$$

式中　$[X]_T$——产物 X 的总浓度。

为了完成热力学模型，需要再引入 4 个系统参数，即每个反应的平衡常数。设反应式（1.1）~式（1.4）的平衡常数分别为 K_{H1}、K_{H2}、K_{HT}和 K_D。由于平衡常数等于反应物浓度与产物浓度的比率，因此，模型中又增加了如下 4 个方程：

$$
\begin{aligned}
K_{H1} &= \frac{[P1][T1]}{[H1]} \\
K_{H2} &= \frac{[P2][T2]}{[H2]} \\
K_{HT} &= \frac{[T1][T2]}{[HT]} \\
K_D &= \frac{[P1][P2]}{[D]}
\end{aligned} \tag{1.6}
$$

退火效率 ε_{ann}（n）用第 n 次退火之后各种杂交产物的量与 n 次退火过程中模板的总量之比来计算，即

$$
\varepsilon_{ann}(n) = 0.5\left(\frac{[H1]}{[T1]_T} + \frac{[H2]}{[T2]_T}\right) \tag{1.7}
$$

其中的［H1］和［H2］可以通过求解上述 8 个方程组成的非线性系统得到，这 8 个方程包含 8 个未知数和 8 个自由参数。

手工求解，或者利用计算机的符号求解法，可以求得该 8 个方程组成的简单系统的解析解。采用本书第 4 章介绍的方法也可以求其数值解。

1.4.2　经颅磁刺激的建模

经颅磁刺激（Transcranial Magnetic Stimulation，TMS）利用电磁感应刺激大脑皮层组织，是临床神经系统诊断和治疗的一种很有潜力的新方法。把线圈放在脑壳表面，磁场就可以穿入大脑组织。TMS 在研究大脑图谱以及治疗大脑疾病方面具有很好的应用前景（Hallett，2000）。TMS 与脑电图（EEG，见第 10 章）等技术一样，其优点是时间分辨率高，缺点是三维空间分辨率以及穿透深度不理想。磁场很难聚焦于大脑中的某个特定点，因此，空间分辨率较差。

Norton（2003）提出了另一种刺激脑组织的方法，可以增加 TMS 的穿透深度，并且提高聚焦能力。他的思路是在强直流磁场下施加超声波来产生电流。研究结果显示，这种方法产生的磁场强度比传统 TMS 要小，但是，其感应磁场的

空间特性和时间特性都有明显的改善。

不难想象，在计算大脑神经组织电场强度时，数值方法起了重要的作用。简而言之，大脑皮层组织用三维柱坐标系统（r, ϕ, z）表示，假设超声波平行于 z 方向传播，并且，其强度 $p(r)$ 是轴对称分布的。则 Norton（2003）建立的微粒速度方程为

$$v(r) = v_0 p(r) \mathrm{e}^{\mathrm{i}k_0 z} \hat{z} \tag{1.8}$$

式中 v_0——微粒的最大速度；

$\hat{z}$——z 方向的单位矢量；

k_0——波数，$k_0 = \omega / c_0$，也就是 k_0 等于超声波频率除以超声波速度。

脑组织中感应磁场 $\boldsymbol{E}_s$ 的各个分量为

$$\begin{aligned} E_r(r,\phi,z) &= \boldsymbol{B}_0 v_0 \left[\frac{\mathrm{d}^2 A(r)}{\mathrm{d}r^2}\right] \mathrm{e}^{\mathrm{i}k_0 z} \sin\phi \\ E_\phi(r,\phi,z) &= \boldsymbol{B}_0 v_0 \left[\frac{1}{r}\frac{\mathrm{d}A(r)}{\mathrm{d}r}\right] \mathrm{e}^{\mathrm{i}k_0 z} \cos\phi \\ E_z(r,\phi,z) &= \boldsymbol{B}_0 v_0 \left[\mathrm{i}k_0 \frac{\mathrm{d}A(r)}{\mathrm{d}r}\right] \mathrm{e}^{\mathrm{i}k_0 z} \sin\phi \end{aligned} \tag{1.9}$$

其中

$$A(r) = K_0(K_0 r)\int_0^r I_0(K_0 r')p(r')r'\mathrm{d}r' + I_0(k_0 r)\int_r^\infty K_0(k_0 r')p(r')r'\mathrm{d}r' \tag{1.10}$$

这里的 $I_0(\cdot)$ 和 $K_0(\cdot)$ 都是超声波传播的仿真函数。

该数学模型用于估计超声波产生的大脑感应磁场的分布，方程组需要用数值方法求解。因为要求解 3 个磁场分量方程，必须先求解式（1.10），获得 $A(r)$。除了极其理想的情况之外，式（1.10）的积分方程和式（1.9）的微分方程组分别要用第 6 章介绍的数值积分和第 7 章介绍的数值微分方程求解器进行计算。

1.4.3 心脏电生理建模

在美国以及其他国家，心率失常引起的死亡都排在因病致死的前列。现在，计算机仿真已成为模拟这些致命状态病因的有用工具（见第 10 章）。高准确度的仿真需要细致密集的空间采样点以及毫秒级的时间步长分辨率。更复杂的是，还要模拟心脏传输、血流以及电活动等很多因素。并且，心脏的几何结构并不简单，构成心脏的组织也不止一种。因此，完整的心脏仿真需要快速计算机处理器以及大容量内存。

本书第10章举例说明了心脏血流及其传输的建模仿真。Pormann等人(2000)开发了一个描述心脏电流活动的仿真系统，该仿真基于一组偏微分方程，称为Bidomain方程，是心脏电生理仿真的常用模型，即

$$
\begin{aligned}
\nabla\sigma_i\ \nabla\Phi_i &= \beta C_m \frac{\partial V_m}{\partial t} + I_{ion}(V_m, q) \\
\nabla\sigma_e\ \nabla\Phi_e &= -\beta C_m \frac{\partial V_m}{\partial t} - I_{ion}(V_m, q) \\
\frac{\mathrm{d}q}{\mathrm{d}t} &= M(V_m q)
\end{aligned}
\tag{1.11}
$$

式中　Φ_i，Φ_e——细胞内和细胞外的电位；

V_m——跨膜电位，即 $V_m = \Phi_i - \Phi_e$；

q——一组状态变量，用于定义细胞结构的生理状态；

I_{ion}，M——细胞膜动力学特性的近似函数；

C_m——跨膜电容；

σ_i，σ_e——电导率。

这组系统方程可以用于模拟一维神经纤维、二维组织薄片或者三维心脏结构。模拟坏死组织或病态组织时，σ_i 可以是不均一的。模拟神经、心房或心室动力学过程时，所用的 I_{ion} 函数和 M 函数各不相同。模拟病态组织或者研究某种通道阻断药物对于电导率的影响时，模型参数可以随空间位置的不同而变化。

根据不同的问题，求解Bidomain方程可以得到一维、二维或者三维空间中各个点的 V_m 电位。Pormann等人(2000)论文中提供了10种数值积分方法，用于求解这组偏微分方程。其中的许多算法，如显式积分法、半隐式和全隐式积分法、自适应时间步长法以及龙格-库塔法(Runge-Kutta Method)等，将在本书第7章讲述。

1.4.4 应用数值方法模拟心血管系统对于重力作用的反应

航天计划开始实施以后，宇航员遇到的问题之一就是心血管系统在返回正常重力环境时所产生的反应。回到地球的宇航员会表现出航天飞行后直立耐力不良(Orthostatic Intolerance，OI)现象，也就是回到正常重力环境之后，不能站立。有关OI的仿真研究很活跃，其中包括：解释航天飞行过程中的各种现象，模拟地球重力环境中各种实验条件下心血管系统的反应，并且，模拟各种调节这些心血管系统问题的方法。Melchior等人(1992)的论文是一篇综述，总结了数种OI仿真模型。

模拟重力反应最好的例子是Heldt等人(2002)的工作，他们用单个心血管模型模拟了地面测试的稳态反应和暂态反应，并且，将模拟结果与测试对象的总

体平均血流动力学数据进行了比较，发现两者吻合得很好。他们的模型为解释各种实验现象、研究各种有关 OI 生理起因的假说提供了一种方法。

血流动力学用电路模型模拟，如图 1.2 所示为一个单房室模型。假设电路元器件都是线性的，模型可以用一阶微分方程组描述。虽然方程用的是电学参量，但是，在线性条件下，该模型可以用于血流动力学的仿真。在各个血压 P 的作用下，流过各个阻抗 R 以及血管顺应性 C 的不同血流速率 q 由如下方程组给出：

$$
\begin{aligned}
q_1 &= (P_{n-1} - P_n)/R_n \\
q_2 &= (P_n - P_{n+1})/R_{n+1} \\
q_3 &= \frac{\mathrm{d}}{\mathrm{d}t}[C_n(P_n - P_{\text{bias}})]
\end{aligned}
\tag{1.12}
$$

各参数下标的定义如图 1.2 所示。

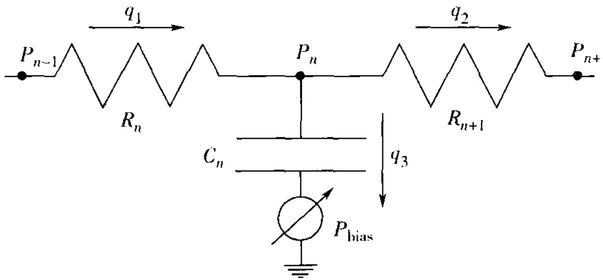


图 1.2　单房室的电路模型示意图

注：P—血压（即电压）　R—阻抗（即电阻）　C—血管顺应性（即电容）

q_1、q_2 和 q_3—血流速率（即电流）　$n-1$、n 和 $n+1$—各个房室的下标

P_{bias}—外部血压。取自 Heldt 等人（2002）的论文。

在节点 P_n 处应用电荷守恒定律，可得 $q_1 = q_2 + q_3$。守恒定律的速率形式为

$$
\frac{\mathrm{d}P_n}{\mathrm{d}t} = \frac{P_{n+1} - P_n}{C_n P_{n+1}} + \frac{P_{n-1} - P_n}{C_n R_n} + \frac{P_{\text{bias}} - P_n}{C_n}\left(\frac{\mathrm{d}C_n}{\mathrm{d}t}\right) + \frac{\mathrm{d}P_{\text{bias}}}{\mathrm{d}t}
\tag{1.13}
$$

图 1.3 显示了整个人体心血管系统的房室模型。外周循环分为上身、肾脏、内脏以及下肢几个部分，上腔静脉、下腔静脉和腹腔静脉作为独立部分，于是，整个模型有 10 个房室，每个房室都用线性电阻 R 和电容 C 表示。该模型与第 10 章介绍的 PHYSBE 心血管系统有许多相似之处，但 PHYSBE 模型更详细。

Heldt 等人的模型用了与式（1.13）类似的 12 个微分方程，并采用自适应步长的 4 阶龙格-库塔积分程序（见第 7 章）求解微分方程组的数值解。根据报道，他们所用的积分步长范围为 $6.1\times10^{-4}\sim0.01$s，平均步长为 5.6×10^{-3}s。先假定所有血压恒定不变，求解血流动力学系统的线性代数方程，得到各个初始血

压值，再用于求解微分方程组。这些方法将在第 4 章讲述。

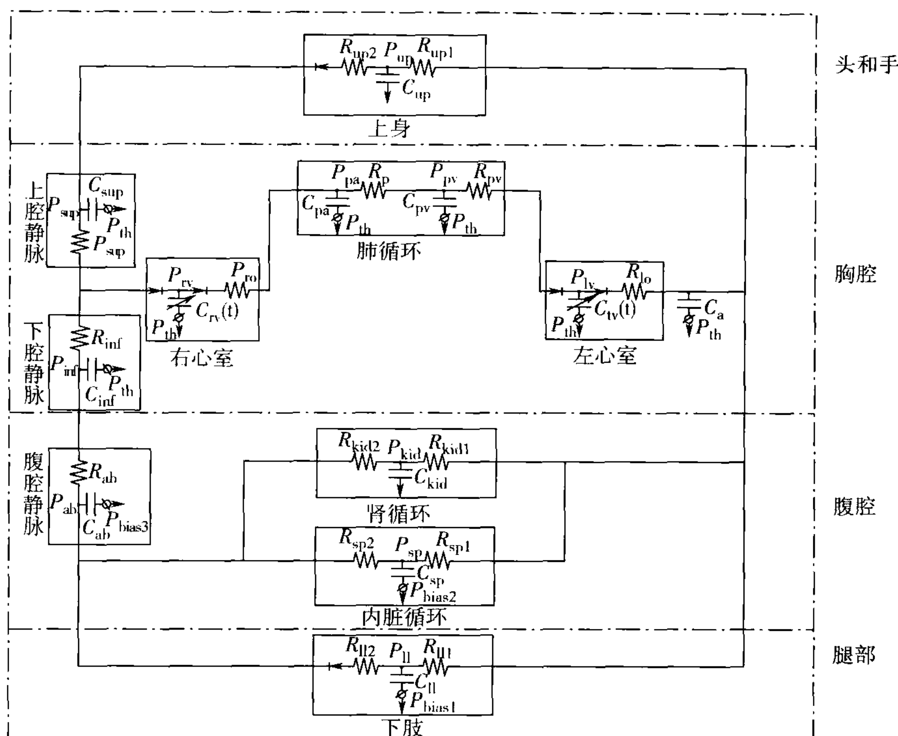


图 1.3　血流动力学系统的电路示意图

注：各个下标的含义为：lv—左心室　a—动脉　up—上身　kid—肾脏　sp—腹腔　ll—下肢　ab—腹腔静脉　inf—下腔静脉　sup—上腔静脉　rv—右心室　p—肺　pa—肺动脉　pv—肺静脉　ro—右心室输出　lo—左心室输出　th—胸腔　bias—同图 1.2 中的定义。取自 Heldt 等人（2002）的论文。

1.5　本书概况

本书介绍如何应用各种计算原理和计算方法求解生物医学工程领域中出现的各种问题，目标是使学生们掌握运用数值方法的知识，也就是能够使用各种数值方法求解生物医学工程问题，注重于基础知识的理解和应用。并且，使读者可以广泛接触各种原理和方法。但是，不讲述各种算法的严格理论推导，这些数学理论基础应该是有关数值方法的数学课程和计算机课程所要讲述的内容。

本书的另一个目标是给读者列举各种建立生物医学系统工程模型的方法。这些示例中的模型都体现了本章所述的守恒定律这个主题。整本书的内容分成四大部分：基础知识、稳态行为模型的应用、有限时间行为模型的应用以及暂态行为

模型的应用。书中采用了生物医学工程各个不同研究领域的实例，包括生物医学仪器、图像学、生物信息学、生物力学以及生物材料等。

1.5.1 第1部分：基础知识

本书第1部分介绍数值方法的基本原理。数值方法必须编程实现，本书的示例和习题等都用 MATLAB 语言编程，因此，这部分首先简要介绍 MATLAB 语言程序开发的基本术语和规则。读者也可以用其他计算机语言实现数值算法，所以，这里也介绍一些计算机科学中的通用概念，如框图设计、数据结构、算法分析等。重点是算法实现的设计及其得失分析。

这部分还包括数字表示法，以及数字表示对于计算结果的准确度、精度和稳定性的影响。在生物医学和医疗保健领域，保证准确度和稳定性，使系统尽可能可靠，这是最基本的要求。无论是否使用 MATLAB 语言，记住计算机的某些基本特性，例如机器可以表示的最小数值 ε 等，有利于编程人员了解程序可靠性分析、系统设计和传播误差控制的重要性。

最后，这部分介绍了数值分析中最重要的一个概念，这就是在连续模型转化为离散模型及其求解算法中，泰勒级数近似所起的作用。在推导数值算法、估计离散化近似所引入的误差、分析系列计算之后误差的传播等方面，泰勒级数都具有重要的作用。

1.5.2 第2部分：系统的稳态行为（代数模型）

本书第2部分概述稳态系统的分析方法。稳态系统的模型用线性或非线性代数方程描述，如果单个方程的未知数是显式的，那么用中学所学的代数就可以求解；如果方程的未知数是隐式的，就要用求根的方法；如果模型是联立方程组，那么要用数值代数方法求解，当然，这里也包括隐式方程组的求解。本书将一一讲解这些方法。

1.5.3 第3部分：系统的动态行为（微分方程）

本书第3部分是生物医学工程人员最感兴趣的内容，就是建模仿真动态系统的暂态行为，并求解系统的输出。包括应用数值方法以及 Simulink 仿真工具求解常微分方程和偏微分方程（附录 B 中有 Simulink 的指南）。

这部分还包括用积分方程模拟的有限时间段上的系统行为仿真。这种情况下，要用数值积分和数值微分求解模型的输出参数。在介绍这些方法的同时也讲述提高计算结果准确度的方法。这部分反复强调的一个主题是要在计算结果的准确度与计算量之间权衡得失。

1.5.4　第 4 部分：建模工具及其应用

本书第 4 部分介绍复杂系统的建模仿真以及分析复杂系统行为的工具和方法。举例说明循环系统、呼吸系统以及神经系统等多房室模型，并给出了这些计算模型的 MATLAB 实现程序。还介绍统计学和时间序列分析的方法及其应用。

随着问题的不断深入和细化，生物医学系统数学建模仿真的需求不断增加。生物医学领域的工程人员学习和掌握本书的内容之后，将有能力实现算法，求解生物医学系统的数学模型，获得系统的稳态行为、有限时间行为和暂态行为。这些人员将具备分析技能、计算技能，并掌握连接这两者的数值方法。

1.6　本章学习要点

学习本章之后，读者应该了解以下内容：

1）数学模型是生物医学工程人员用于预测系统行为的工具。

2）生物医学系统模拟包括 3 种不同状态：稳态行为、有限时间段内的行为、暂态行为。系统模型可以模拟其中的一种或多种状态。

3）数学模型的建立从守恒定律开始。

4）数值方法是模型的解析方程与计算机程序之间的桥梁。

1.7　习题

1.1　请列举 5 个计算机在生物医学工程领域应用的例子，并作简单说明。

1.2　假设你接到一项任务，要设计一个计算机控制的病人监护系统，用于中等规模社区医院的重症监护病房。你要监测的参数有哪些？计算机将起什么作用？

1.3　为什么经过计算机平均处理之后的脑电图诱发响应比原始信号容易分析？

1.4　计算机给临床实验室的自动化带来了哪些好处？

1.5　请列出医院重症监护病房和重症冠心病监护病房（ICU 和 CCU）的常用监护设备。

1.6　计算机系统是否总是能用于生物医学仪器？

1.7　请列举 3 个计算机的医学研究应用实例。

1.8　在一次生物医学工程课程考试中，要求计算某种人工心脏装置克服重力作用可以将血液泵出的最大距离，学生乔治在试卷上写下这个距离数值时，把英尺数和英寸数颠倒了，结果他写的答案只有计算长度的 30%，设该计算长度小于 10ft，并且英尺数和英寸数都为整数，乔治计算的长度为几英尺几英寸？

1.8 参考文献

Enderle, J. D., Blanchard, S. M., and Bronzino, J. D. 2000. *Introduction to Biomedical Engineering*. San Diego, CA: Academic Press.

Fournier, R. L. 1999. *Basic Transport Phenomena in Biomedical Engineering*. Philadelphia, PA: Taylor & Francis.

Gevertz, J. L., Dunn S. M., Roth, C.M. 2005. Mathematical model of real-time PCR kinetics. *Biotechnol. Bioeng.*, in press.

Hallett. M. (2000). Transcranial Magnetic Stimulation and the Brain. *Nature*, **406:**147-150

Heldt, T., Shim, E. B., Kamm, R. D., Mark, R. G. 2002. Computational modeling of cardiovascular response to orthostatic stress. *J Appl Physiol.*, Mar:**92**(3): 1239-1254.

Melchior, F. M., Srinivasan, R.S., Charles, J.B. 1992. Mathematical modeling of human cardiovascular system for simulation of orthostatic response. *Am J Physiol.*, **262**(6 Pt 2):H1920-1933.

Norton, S. J. 2003. Can ultrasound be used to stimulate nerve tissue? *Biomed. Eng. Online*, **4**:2(1):6.

Nebeker, F. (2002). Golden accomplishments in biomedical engineering. *IEEE Engineering in Medicine and Biology Magazine*, **21**:17-47.

Pormann, J. B., Henriquez, C. S., Board, J. A., Rose, D. J., Harrild, D. M., and Henriquez, A. P. 2000. Computer Simulations of Cardiac Electrophysiology. Article No. 24, *Proceedings of the 2000 ACM/IEEE Conference on Supercomputing*. Dallas, TX.

Thompson, W. J. 2000. *Introduction to Transport Phenomena*. Upper Saddle River, NJ: Prentice Hall PTR.

Vierstraete, A. 1999. http://users.ugent.be/~avierstr/principles/pcr.html. Last viewed August 24, 2005.

第2章 计算技术入门

2.1 绪论

计算机是所有工程领域都要用到的工具，生物医学工程也不例外。在生物医学工程领域，计算机用于仪器设备的控制、数据采集、图像分析、模型仿真以及统计分析等各个方面。从事生物医学工程的人员如果具备良好的计算技能，就可以得心应手地求解该领域中各式各样的问题。

本章介绍计算技术的核心内容：编程语言及程序设计、数据结构以及算法分析。其中列举了许多 MATLAB 程序，说明如何应用程序设计原理开发良好的计算机程序。

本章包括如下学习内容：

1）了解计算机在生物医学工程中的应用。

2）了解命令语言、函数语言和面向对象编程语言的区别。

3）了解计算机编程语言的常用数据结构。

4）数字在计算机编程语言中的表示方法。

5）计算机算法及其在生物医学工程中的应用。

6）了解结构化程序设计语言以及良好的编程方法。

7）编写 MATLAB 程序。

2.2 计算机在生物医学工程中的角色

计算机在医学仪器中的广泛应用使得许多大众消费者可以自己购买和使用医疗仪器，例如，测量血压的血压计（见图 2.1）就是这样一种仪器。

计算机通过以下几个方面为用户操作血压计提供了方便：

1）提供用户界面

2）控制整个系统

3）存储数据

4）提供系统的信号处理功能

5）保证整个系统操作（如袖带充气等）的安全性和可靠性

用于血压（Blood Pressure，BP）监测的计算机程序必须具备所有这些功能，

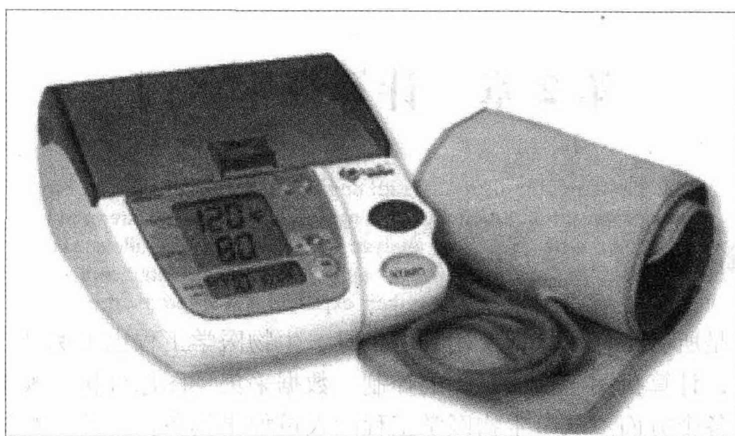


图 2.1 一种家用自动血压计（图片由 Omron 保健公司提供）

并且每次使用时都要正确无误地执行每个功能。因此，设计和编写计算机程序时必须牢记以下准则：

1）程序必须正确无误，每次运行都一样。

2）用户必须确信所有的测量都一样可靠。

3）测量仪必须便于使用。

家用血压计只是计算机在生物医学工程中应用的一个例子。还有医学图像，尤其是计算机断层扫描成像（Computed Tomography，CT）和磁共振成像（Magnetic Resonance Imaging，MRI）；以及生物信息学；血液动力学等的仿真；高分辨率显微镜技术等都是计算机应用的结果。

血压监测仪、CT 和 MRI 设备以及显微镜等仪器都具有一个相同的结构，所有这些仪器设备中，都用计算机控制用户界面和数据采集等电子模块，这些模块与其他用于参数检测和传感器控制的子系统相连。

经过培训，用户就可以操作仪器所有的功能。但是，由于普通消费者无培训，以及人的工作能力会随时间发生变化；因此，似乎不能把血压计等仪器交给普通消费者使用。但是，仪器中安装的计算机已经使操作标准化，未经培训的用户也可以操作血压计等医学仪器，以便更好地掌握自己的健康状况。

给计算机设定一组标准化操作指令之后，每次使用时，仪器都会执行同样的功能，除非硬件出了故障。因此，使用计算机的优点有：

1）计算机产生的动作及其控制程序可以一模一样地重复，除非有电子器件、电动装置或者传感器等发生故障，重复的操作不会随时间变化。

2）可以控制精度和准确度。精度是指事件可能分辨的最小程度，而准确度是指测量值接近真实值的程度，两者的含义是不同的。精度和准确度与所采用的

传感器和计算方法都有关系，本书将着重讲述如何最优化计算机程序，以获得最优的精度和准确度。

计算机也有缺点，在某些方面不如人工控制，例如：

1）计算机需要精确完整的操作指令，不能有任何含糊不清。

2）计算机程序的测试常常不彻底。程序测试，尤其是大型程序的测试，本身就是一项工作。经常会出现没有预料到的输入情况或者事先没估计到的输入冲突。采集的数据可能会比所需要的多。可能的输入情况组合数按照自变量数目的指数增长，即使是中等程度的输入变量数目，可能的输入组合数目就太大了，不可能全部进行测试。

3）程序可能存在逻辑或语法错误，在以后的使用过程中会出错。生物信息学软件之类的大型应用程序设置有特定的网络界面，用于跟踪程序中存在的问题和缺陷。例如，利用 Java 工具处理基因序列之类生物医学数据的网站 http：//www. biojava. org，就是一个很好的例子。

除了控制仪器，使设备实现自动化之外，计算机还给生物医学工程人员提供了以下便利：

1）提供设计工具，包括 Simulink、Pro-E、ANSYS 等仿真工具。

2）使用 MS Word 和 Excel 等工具保存文档。

3）提供测试手段并给出测试结果，例如，MATLAB 和 Labview 等工具。

本书的重点是帮助生物医学工程领域的新手掌握开发计算机程序的方法，用于控制测量和计算的精度及准确度，求解生物医学实际问题。要进行程序设计，首先就必须熟悉 MATLAB 等编程环境具有的功能。

2.3　程序设计语言工具及方法

目前所用的程序设计语言有 3 类：命令类（即状态转换类）、面向对象类以及函数类。每一类都有多种编程语言。那么，应该如何选择程序设计语言呢？

最好的命令类（即状态转换类）语言是 C 语言和 Fortran 语言，用这种命令语言编写的程序按照语句的顺序一条一条执行。

面向对象的编程语言是目前最多、使用最广泛的语言，有 C + + 、Java、Python 和 Smalltalk，最初就是 Smalltalk 掀起了面向对象的程序设计革命。面向对象程序的执行与命令程序的执行很不相同，程序所执行的操作由数据对象及其需要的动作决定，所执行的动作则取决于对象的类型，例如，虽然运算操作的名称可能相同，都是加“add”，但两个整数的求和与两个矢量的求和是不同的。本书的例题都用 MATLAB 语言编写。MATLAB 是面向对象的语言，具有面向对象的编程环境。建议不熟悉 MATLAB 的读者先学习一下附录 A 有关 MATLAB 编程语

言及其开发环境的介绍。

基于函数编程语言的程序由一系列函数定义及其调用构成，函数的调用可以像命令语言那样顺序执行，也可以并行执行。常用的函数编程语言有 Scheme、Sisal 和 Caml 等。函数编程语言往往具有面向对象的性质。本书不使用这类语言。

命令语言和面向对象的语言都有常用的结构化语句，帮助编程人员有效地开发程序、设计算法以及程序执行流程，使程序易于阅读并且易于调试。现在的编程语言限定了 3 种控制流语句，即顺序语句、条件执行语句和循环语句。具备这些结构化语句的语言称为结构化语言。这 3 种结构化语句都只有一个输入和一个输出，使程序的设计和开发标准化。

2.3.1 顺序语句

顺序指令有特定的符号或关键字作为定界符，这些定界符指明了模块的单个起始点和终止点。

例 2.1 顺序语句程序

请用 31 与 75 之间的所有偶数创建一个矢量。

解：

31 与 75 之间的所有偶数就是 32 ~ 74 的偶数。创建这个矢量有两种方法：列出这些数字；用 MATLAB 的省略写法，即“第一个数：增量：最后一个数”。

第一种方法留给读者。第二种方法求解如下：

1）矢量的第一个数为 32；

2）矢量只包含偶数，因此增量为 2；

3）矢量的最后一个数为 74。

于是，矢量可以设为 [32：2：74]。MATLAB 的输出为

```
>> x = [32:2:74]
x =
  Columns 1 through 11
32    34    36    38    40    42    44    46    48    50    52
  Columns 12 through 22
54    56    58    60    62    64    66    68    70    72    74
```

2.3.2 条件执行语句

如果程序的执行路径需要根据外部输入数据的值来确定，就要使用条件语句。条件语句也有定界符（即关键词），通常是 if…then，或 if…then…else。这

种模块也只有关键词 if 指示的一个入口，以及一个出口，就是从分支回到程序控制流的主线。

1. If…then 语句

最简单的条件执行语句是 if…then 语句，在 MATLAB 中的形式是“if 表达式 语句 end”。如果其中的“表达式”成立，则执行其中的“语句”。“表达式”肯定要计算，控制流也一定要回到紧跟“end”之后的语句上。

2. If…then…else 语句

if…then 语句只提供一种情况的条件执行控制流，而 if…then…else 语句则提供两种情况，控制流只执行其中之一，不会两种情况都执行。

在 MATLAB 中，if…then…else 语句的语法为

```
if  表达式
  语句 1
else
  语句 2
end
```

“表达式”总是要计算的。如果“表达式”成立，则执行“语句 1”模块；如果“表达式”不成立，就执行“语句 2”模块。

MATLAB 还有一种条件执行语句，就是 if…elseif…else…end 语句，其语法是

```
if  表达式 1
        语句 1
elseif  表达式 2
        语句 2
else
        语句 3
end
```

其中的“elseif”子句增加了一个表达式的计算。如果第一个表达式不成立而第二个表达式成立，就执行“语句 2”模块。下面举例说明其使用方法。

例 2.2　使用 if…then…else 语句的简单控制流

请编写一个 MATLAB 脚本，计算以下函数：

$$f(x) = \begin{cases} -1 & x < 0 \\ 0 & x = 0 \\ 1 & x > 0 \end{cases}$$

解:

题目要求建立一个 MATLAB 脚本文件（即 m 文件），计算如上定义的 f 函数，脚本根据不同的输入数据，返回 3 个值中的一个。这个数学函数很容易转化成 MATLAB 指令，算法分析如下：

如果参数 x 小于 0，m 文件就返回 -1，将其写成 MATLAB 语句就是

```
if x<0  f=-1;
```

其中的变量 f 就是函数返回值的变量。

如果 x 小于 0，则返回值已获得（即 f = -1），就不需要执行其他语句了。如果 x 的值不小于 0，那么 f = -1 不执行，f 的值待定。

如果 x 的值不小于 0，则要么等于 0，要么大于 0。根据如上数学定义，如果 x 等于 0，则 f 的值应为 0；如果 x 大于 0，则 f 的值应为 1。

计算这个函数的 MATLAB 脚本需要 3 个“表达式—语句”对，这 3 个语句不能依次顺序执行。若执行了第一个语句，函数 f 的计算就应该结束。知道为什么吗?

为了测试程序执行流程，可以设置各种不同情况，考察程序的执行过程。

完成该算法的控制结构需要用 elseif 子句，最终的程序为

```
if x<0
    f=-1;
elseif  x==0
    f=0;
elseif  x>0
    f=1;
end
```

补充说明一下，这段程序的执行结果与 MATLAB 的内置函数 sign 相同，读者可以验证。

3. switch 语句

大多数模块化编程语言（包括 MATLAB）可以根据条件从任意多个选择中选定某段程序执行。当然，利用 if…elseif…end 语句就可以实现这种控制流，但是，MATLAB 有更方便的方式，就是 switch…case…otherwise…end 语句。switch 语句的语法为

```
switch switch_表达式
    case case_表达式
```

```
        语句, …, 语句
    case { case _表达式 1, case _表达式 2, case _表达式 3, …,}
        语句, …, 语句
    ...
    otherwise
        语句, …, 语句
end
```

其中，每种选择对应一条 case 语句。case 语句由 3 部分组成：关键词 case、一个或多个 “case _表达式”、以及一条或多条语句组成的语句段。当 “switch _表达式” 的值与某个 “case _表达式” 的值相等时，就执行相应的 case 语句段。如果没有 “case _表达式” 与 “switch _表达式” 相等，就执行 “otherwise” 中的语句。

一旦语句被执行之后，与所有其他结构语句一样，控制流就回到紧跟 switch 语句之后的语句上。

例 2.3　switch 语句的使用

编写 MATLAB 脚本，读取一个字符变量（*s* 或 *c*）和一个整数变量（1、2 或 3），根据以下表格的内容，作出相应的函数曲线。

	1	2	3
s	$\sin\theta$	$e^{-\theta}\sin^2\theta$	$\sin3\theta/e^{-\theta}$
c	$\cos\theta$	$\cos2\theta/e^{-2\theta}$	$e^{-\theta}\cos^3\theta$

假定作图区间 $0\leq\theta\leq 2\pi$ 上有 100 个点。

解：

在开始编写本题的 MATLAB 程序之前，最好先熟悉一下作图函数的用法。plot 作图函数有很多种不同的形式，其中，plot（***Y***）是以下标为自变量作出 ***Y*** 矢量的曲线；plot（***X***，***Y***）则是将 ***X*** 矢量作为自变量，将 ***Y*** 矢量作为因变量作出的曲线图。例如，图 2.2 显示的是以下 MATLAB 指令的运行结果：

```
>> x = 1:100;
>> plot (sin (x/3)) ;
```

注意，图 2.2 中自变量 *x* 的值作为水平轴，因变量 *y* 的值作为垂直轴。

本例题的自变量是 θ 角，单位是弧度。由于题中设定作图范围内有 100 个值，自变量的取值即为

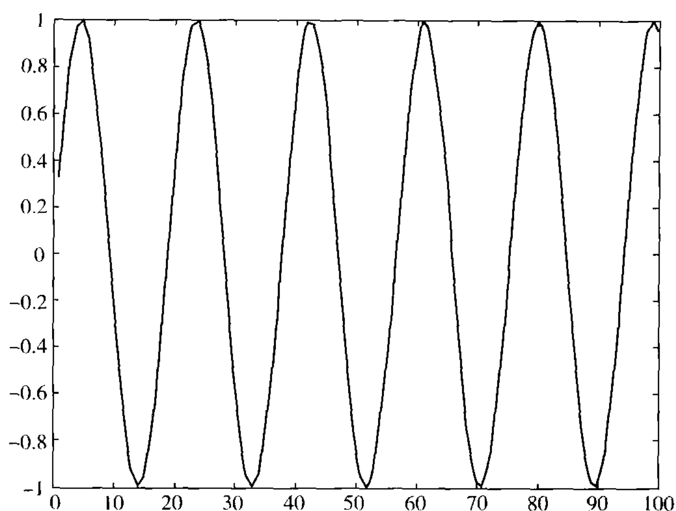


图 2.2 sin（x/3）曲线

```
x =0:(2 * pi)/100:2 * pi
```

题中还设定 MATLAB 脚本需要读取 2 个输入数据，这 2 个输入值分别作为如上表格的行和列的标志，行标志用字符 "s" 表示正弦函数，用字符 "c" 表示余弦函数；列标志则用整数 1、2 和 3。MATLAB 脚本要根据这 2 个输入值，选定表格中的函数，计算该函数的值并作出其曲线。

用 switch 语句实现本题的程序流控制最方便，如果用大量的 if-then-else 语句会比较麻烦。简捷的算法是分别用 switch 语句实现行的选择和列的选择。开始时，可以写成

```
switch letter
case  's'
        这里加入"s"行的 switch 语句
case  'c'
        这里加入"c"行的 switch 语句
end
```

这样，就可以独立编写和调试各行的 switch 语句，遵循了良好的程序设计原则。其中，"s" 行的 switch 语句为

```
switch col
          case 1
```

```
            plot(theta,sin(theta));
        case 2
            plot(theta,exp(-theta).*(sin(theta)).^2);
        case 3
            plot(theta, sin(3*theta)./exp(-theta));
        end
```

这就是 case 's'下要插入的模块。

本题完整的 MATLAB 脚本为

```
twopi=2*pi;
theta=0:(twopi/100):twopi;
letter=input('Please enter either s or c:  ');
col=input('Please enter either 1, 2 or 3:  ');
figure(1);
switch letter
case  's'
        switch col
        case 1
            plot(theta,sin(theta));
        case 2
            plot(theta, exp(-theta).*(sin(theta)).^2);
        case 3
            plot(theta,sin(3*theta)./exp(-theta));
        end
case  'c'
        switch col
        case 1
            plot(theta,cos(theta));
        case 2
            plot (theta, cos (2*theta)./exp (-2*theta));
        case 3
            plot(theta,exp(-theta).*(cos(theta)).^3);
        end
end
```

下面是该脚本的某次运行过程，图 2.3 显示了其结果。考虑一下，为什么 plot 函数输出图形上横坐标（自变量）的值是 0 ~ 7？

```
Please enter either s or c:  's'
Please enter either 1, 2 or 3:  2
```

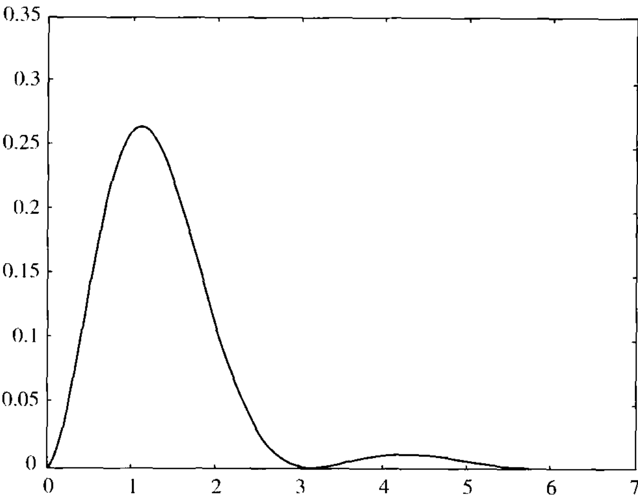


图 2.3

2.3.3 循环语句

1. while 循环

循环指的是某个指令模块被反复执行，直到满足某个条件为止。这个条件可以是数据的状态，也可以是外部输入。一种通用的循环结构是 while 语句。

例 2.4 while 循环的使用

请编写一个脚本，求随机数之和达到 20 时所需要的随机数的数目。本题要用到 MATLAB 的随机数发生器 rand。

解：

MATLAB 的随机数发生器 rand 产生 0 ~ 1 之间均匀分布的一个伪随机数。之所以称为伪随机数，是因为 rand 产生的数值与其被调用时计算机的状态有关。例如，以下调用

```
>> rand
ans =
    0.9501
```

在同样的运行情况下，总是产生这个数。rand 函数返回的数大于或等于 0，但必定小于 1。

本题的要求是：一旦产生的随机数之和达到 20，MATLAB 脚本就要停止执行。也就是，脚本要不断地重复产生随机数，把新的随机数加入总和，直到总和大于或等于 20 为止。在此过程中，还要计算随机数的个数。因为随机数的个数不可能事先设定，所以，程序只能重复产生随机数，直到满足条件为止。这里用 MATLAB 的 while…end 结构来控制循环。解题的算法如下：

1）随机数产生之前，随机数总和的初始值必须设为 0，计数变量也设为 0；

2）检查随机数总和是否大于或等于 20，如果是，则结束脚本；否则继续第 3 步；

3）产生一个随机数，将其加入总和，并将计数变量增 1，再返回到第 2 步。

MATLAB 脚本如下，保存为 whileloop. m 文件：

```
% whileloop.m
add =0;
count =0;
while add <20
    add = add + rand;
    count = count +1;
end
display(count);
display(add);
```

在指令窗中键入该脚本名，执行程序并得到如下输出结果：

```
>> whileloop
count =
    43
add =
    20.0364
```

2. for 循环

与条件控制结构一样，循环控制结构也有好几种。for 循环结构（MATLAB 中是 for…end 语句）的循环次数一定，通常由控制变量的值指定，例如读取数组中不同位置的元素，这个操作就可以用 for 循环实现。

例 2.5 for…end 循环的使用

给定矢量 $\boldsymbol{x}=[1\ 8\ 3\ 9\ 0\ 1]$，编写 MATLAB 脚本，求矢量所有元素之和，并用 sum 指令验证结果的正确性。

解：

由题目得知，脚本求解的对象是一个已经给定的矢量，不是任意矢量。矢量的长度已知，为 6，因此，脚本的一种写法是

```
x=[1  8  3  9  0  1];
add=0;
add=add+x(1);
add=add+x(2);
add=add+x(3);
add=add+x(4);
add=add+x(5);
add=add+x(6);
add
sum(x)
```

注意，这组指令中有个相同的部分，也就是将数组每个元素加入总和的操作。如果矢量再长一点，要写出这样的指令组就很乏味了。程序中相同的语句可以用如下 for 循环代替：

```
% forloop.m
x=[1  8  3  9  0  1];
add=0;
for i=1:length(x)
    add=add+x(i);
end
add
sum(x)
```

其中的 for…end 循环简化了 x 元素依次加和的 6 条语句。forloop. m 的执行结果为：

```
>> forloop
add =
22
ans =
    22
```

注意，整个 forloop.m 脚本里的所有语句等价于一条 sum(x)指令，这也体现了 MATLAB 的功能强大。

2.3.4 封装

要编写结构良好的 MATLAB 程序，使程序便于阅读、理解和维护，其最后的关键点是封装。封装就是把有关的指令和语句组织起来，形成特定的函数。封装可以使程序便于阅读、容易调试。如果不封装，MATLAB 程序很长时，就很难处理。

MATLAB 的封装用 m 文件来实现。m 文件是包含 MATLAB 语言程序的文本文件。结构良好的大型 MATLAB 程序，其主程序是一个 m 文件，再加上其他函数和脚本的 m 文件。这些 m 文件可以由主程序调用，也可以在 MATLAB 指令提示符下使用。

现在的编程语言都有程序封装功能，通常有子程序和函数两种封装结构。

MATLAB 脚本是一组指令序列，它不能接收输入参数，也不能返回输出参数。脚本可以使用和修改工作空间中存在的数据。在其他模块结构的编程语言中，MATLAB 脚本这样形式的程序称为子程序。

函数与脚本类似，也是一组指令序列。但是，函数可以接收输入参数，也可以返回输出参数。函数还可以拥有只能在函数内部使用的 MATLAB 变量。

所有模块结构的编程语言中，子程序和函数都只有一个控制流入口以及一个出口。

例 2.6 脚本和函数的使用

编写一个 MATLAB 脚本，调用一个函数，将输入的华氏温度值转换成对应的摄氏温度值。要求脚本持续运行，直到再也没有数据需要转化为止（注意，这里要使用 isempty 指令）。

解：

本题的程序分两部分，一部分是将华氏温度转化为摄氏温度的函数，另一部分是调用该函数的脚本。最好的设计方法是先编写和调试函数，确认其运行正确之后，再编写调用此函数的脚本。

MATLAB 函数用如下转化公式计算摄氏温度

$$C = \frac{5}{9}(F - 32)$$

华氏温度 F 是函数的输入参数，函数返回摄氏温度 C。该函数的程序为

```
% Far2Cel.m
function C = Far2Cel(F)
C = (5/9) * (F - 32)
end
```

在继续下面的编程之前，应先测试该函数。

调用此函数的脚本需要完成的任务是：从键盘读入华氏温度，显示转化之后得到的摄氏温度，然后等待下一个输入。

可见，由于脚本程序段重复执行的次数不定，没有预定值，因此，这里要用 while 循环，而不是 for 循环。

首先，读取键盘输入的华氏温度，检查是否是空输入。如果是，则结束程序的运行；否则，将温度转化为摄氏温度，然后读取下一个输入温度，再检查输入是否为空。重复此过程，直到没有键盘输入为止。脚本程序为 F2C. m：

```
% F2C.m
F = input('The value of the temperature in degree Fahrenheit = ');
while ~ isempty(F)
C = Far2Cel(F);
display(C);
F = input('The value of the temperature in degree Fahrenheit = ');
end
```

在 MATLAB 指令行可以运行这个脚本，下面是程序执行的一个过程。注意，没有数字输入后，程序停止运行。

```
>> F2C
The value of the temperature in degree Fahrenheit = 32
C =
     0
The value of the temperature in degree Fahrenheit = 100
C =
   37.7778
The value of the temperature in degree Fahrenheit = 212
C =
    100
The value of the temperature in degree Fahrenheit = 77
C =
     25
The value of the temperature in degree Fahrenheit =
```

2.4　MATLAB 的数据结构基础

MATLAB 程序由两部分组成：执行算法的指令序列和被处理的数据。上节概述了 MATLAB 的基本编程结构，本节将介绍 MATLAB 的数据表达方式。

虽然本书的重点是数值方法和定量分析，但是，也需要了解变量名、输出结果及其解释等内容。数据输出的格式也很重要。

现在的程序设计语言都可以建立和使用多种不同的数据类型，由基本数据类型可以形成数据集合。本节讲述 MATLAB 的 6 种基本数据类型，即 double、char、sparse、uint8、cell 和 struct。

2.4.1　数的表示

无论是否有特意说明，微积分和微分方程等课程中使用的都是实数。实数具有无限的精度，这在计算机程序中无法实现。无限的精度需要无限的存储器，而计算机的存储器总是有限的。

这样，实数在计算机中就用所谓的浮点数来近似表示。浮点数只用有限的空间存储，这就意味着有的实数不能用浮点数表示。有关这方面的内容会在第 3 章详细讲述。浮点数表示方法对于计算的准确度和精度都有很大的影响，在深入学习数值方法之前必须掌握这方面的内容。

MATLAB 脚本和函数中所用的数使用常规表示法，MATLAB 不区分整数、实数和复数，这些数都用同样的 MATLAB 变量存储。

数字之前可以有加号或者减号，数字中也可以有小数点。科学表示法的格式使用字母 e 表示指数（基数为 10）。虚数则用后缀 i 或者 j 指示其虚部。下面给出了一些 MATLAB 表示数的例子。

例 2.7　MATLAB 中数的表示

正整数	3
负整数	-45
实数	0.00001
科学表示法	2.71828e9
复数	3+5j
虚数	-3.14159j

每个数所占有的存储空间有限，因此，MATLAB 浮点数表示的实数有上限和下限。用 MATLAB 函数 realmin 和 realmax 可以分别返回 MATLAB 所能表示的最小实数和最大实数。函数 inf 返回无穷大的表示，函数 NaN 则返回“不是一个

数”（Not-a-Number）的表示“NaN”，例如，0.0/0.0 之类的无定义运算在 MATLAB 中的运算结果就是“NaN”。

复数可以直接赋给数值变量，也可以由两个实数创建。MATLAB 指令 c = complex(3, 5)就是创建一个复数 c，其值等于 3 +5i。指令 c = complex(3, 0)创建的是一个虚部为 0 的复数，而不是实数。

例 2.8 复数

求复数 3 +5i 的倒数，并手工验证其正确性。

解：

回忆一下，复数 $a+bi$ 的倒数可以用下式计算：

$$(a+b\mathrm{i})^{-1}=\frac{(a-b\mathrm{i})}{(a+b\mathrm{i})(a-b\mathrm{i})}$$

MATLAB 函数 inv（ ）可以求其输入参数的倒数，指令为

```
>> inv(3 +5i)
ans =
    0.0882 -0.1471i
```

用以上公式可以验算此结果，也可以用如下求共轭复数的 MATLAB 函数 conj（ ）来验算结果：

```
>> x =3 +5i;
>> conj(x)/(x * conj(x))
ans =
    0.0882 -0.1471i
```

2.4.2 数组

数组是表示对象集合的一种数据结构，其特点是访问每个集合元素所需的时间相同。要读取其中某个元素，不需要沿着数据列表或数据阵列搜索。

MATLAB 中的所有数组都以列矢量形式存储，用单个列矢量下标就可以访问每个元素。但是，如果数组是多维的，则列矢量中元素的位置就要用如下公式计算：

$$(j-1)*d+i$$

用此下标访问的数组元素是 $A(i,j)$，其中 d 为每列的长度。多维数组的子数组也可以整个提取出来。

例 2.9 MATLAB 数组的访问

建立一个 5 ×5 矩阵，元素的值为 $A(i,j) = 1/(2^i + 3^j)$。输出整个矩阵、矩阵的第一行以及矩阵的第二列。并给出元素 $A(3,2)$ 的两种显示方法。

解：

与许多其他程序设计语言不同，MATLAB 没有数值维数声明等方法用于说明分配给变量的内存空间，MATLAB 自动给矩阵等数组分配内存空间。本题中，创建矩阵 **A** 的元素时就分配了内存空间，该矩阵元素的值由计算公式给定，因此，可以将创建矩阵以及元素输出的语句放在同一个脚本中。矩阵的大小已确定，可以用 for 循环计算元素的值，然后再加上元素的输出语句。MATLAB 脚本为

```
% ArraysRef.m
for i =1:5
   for j =1:5
      A(i,j) =1/(2^i +3^j);
   end
end
A
A(1,:)
A(:,2)
A(8)
A(3,2)
```

注意，数组的第一行用 A (1, :) 表示，下标 1 指行号，第二个下标不是数，是“:”，指该行中的所有元素。A (:, 2) 中的“:”具有同样的含义，只是这里指的是行，指第 2 列的所有行。

记住，MATLAB 的数组都以列矢量的形式保存，按照顺序，一列紧跟一列。因此，元素 A (3, 2) 可以看成一个矢量中的元素，也可以看成矩阵中的元素。

在指令窗中运行以上脚本产生如下结果：

```
>> ArrayRefs
A =
    0.2000    0.0909    0.0345    0.0120    0.0041
    0.1429    0.0769    0.0323    0.0118    0.0040
    0.0909    0.0588    0.0286    0.0112    0.0040
    0.0526    0.0400    0.0233    0.0103    0.0039
    0.0286    0.0244    0.0169    0.0088    0.0036
```

```
ans =
    0.2000    0.0909    0.0345    0.0120    0.0041
ans =
    0.0909
    0.0769
    0.0588
    0.0400
    0.0244
ans =
    0.0588
ans =
    0.0588
```

注意，数组的行显示出来的是一行，列显示的是一列。

2.4.3 字符和字符串

MATLAB 程序的字符表示方法与其他程序设计语言一样，都使用 ASCII 码字符集。其中，32 ~ 127 之间的数分别编码数字、大写字母和小写字母，0 ~ 31 之间的数代表退格键等特殊控制字符。MATLAB 的字符串就是字符的数组。虽然这种数组不能进行数学运算，但是，字符串有自己的运算，如：字符串的连接、字符或字符串之间的比较，以及字符串的搜索、字符串的变换等。

例 2.10 字符串是数组

请分别建立两个字符串变量“Biomedical”和“Engineering”，连接这两个字符串，输出合成字符串，并且输出其中所有 e 字母的下标。

解：

先把两个字符串分配给独立的两个 MATLAB 变量，再用 MATLAB 函数 strcat 把它们连接起来，然后求得连接之后字符串的总长度，并且，将字符串中的每个字符与“e”字母比较，搜索出所有字母“e”的位置。由于比较的次数确定，就是字符串的长度，因此可以使用 for 循环。MATLAB 程序为

```
% Strings.m
B = 'Biomedical';
E = 'Engineering';
BME = strcat(B,E)
L = length(BME);
for i = 1:L
```

```
    if(BME(i)=='e')
        display(i);
    end
end
```

下面是脚本 Strings. m 运行的结果：

```
>> Strings
BME =
BiomedicalEngineering
i =
     5
i =
     16
i =
     17
```

注意，第一，变量 BME 中 Biomedical 和 Engineering 两个字符串中间没有空格，变量 B 和变量 E 在开头和结尾处都没有空格。第二，if 语句的表达式显示了字符串数组的处理方式，字符串中的每个字符可以看作是数组的元素，用 BME（i）访问，与数值数组元素的访问方法相同。如果把字符与一个数进行比较，则数组元素 BME（i）就作为相应的 ASCII 代码处理，不再是字符。第三，大写字母 E 没有被搜索出来，因为比较的只是小写字母 e。

2.4.4　逻辑（布尔）数据类型

MATLAB 没有特定的逻辑（布尔）数据类型，逻辑假 F 就用数值 0 表示，任何其他非 0 数值都表示逻辑真 T。MATLAB 有 3 种逻辑运算，即："~" 表示的一元运算 "非（NOT）"、"&" 表示的两元运算 "与（AND）"，以及 "|" 表示的 "或（OR）" 运算。异或运算则用 MATLAB 函数 xor（）实现，如 xor（A，B）。

逻辑运算常用作条件语句或循环语句的表达式。MATLAB 还有另一种逻辑运算，称为逻辑索引，是从数组中选出满足特定条件元素的一种方法，产生一个索引数组，其中的元素值不是 0 就是 1，1 表示元素被选中。索引数组可以用 MATLAB 的 logical（）函数产生，logical（）函数的输入参数是一个数值数组，返回的是该数组的索引数组。

索引数组作为原数组的下标使用，得到的结果是原数组中满足特定条件的那

些元素。

例 2.11 MATLAB 的逻辑索引

设 $x=1:10$，$y=[3\ 1\ 5\ 6\ 8\ 2\ 9\ 4\ 7\ 0]$，请运行以下指令并解释其输出结果：

```
a. (x>3) & (x<8)
b. x(x>5)
c. y(x<=4)
d. x( (x<2) | (x>=8) )
e. y( (x<2) | (x>=8) )
f. x(y<0)
```

解：

首先建立一个名为 LogicalIndexing. m 的 MATLAB 脚本，用两条语句初始化变量 x 和变量 y，紧跟着写出以上 a ~ f 6 个表达式。然后运行脚本，得到如下输出结果：

```
>> LogicalIndexing
x =
    1    2    3    4    5    6    7    8    9    10
y =
    3    1    5    6    8    2    9    4    7    0
ans =
    0    0    0    1    1    1    1    0    0    0
ans =
    6    7    8    9    10
ans =
    3    1    5    6
ans =
    1    8    9    10
ans =
    3    4    7    0
ans =
  Empty matrix: 1-by-0
>>
```

表达式 a 有两项，第一项是（$x>3$），运算之后返回一个前 3 个元素为 0 的逻辑

数组；第二项是 $(x<8)$，该运算返回一个最后 3 个元素均为 0 的逻辑数组。于是，整个表达式 $(x>3)\&(x<8)$ 输出的逻辑数组就是这两个逻辑数组对应元素进行逻辑“与”运算的结果。由于“与”运算的两个数组中，只有第 4 ~ 第 7 四个元素的值都为“真”，因此，最终的输出数组中只有这 4 个元素为 1，其他元素均为 0。

表达式 b 中的 $(x>5)$ 是逻辑索引数组，x 的前 5 个元素小于或等于 5，因此索引数组的前 5 个元素为 0。用这个矢量作为 x 数组的索引数组，结果只返回索引值为 1 的那些元素，不包含索引值为 0 的元素。因此表达式 b 的输出为 [6 7 8 9 10]。表达式 c 的运算过程与表达式 b 相似。

表达式 d 和 e 的运算过程也与表达式 b 和 c 类似，只是其中求取逻辑索引数组的表达式有两项，这与表达式 a 类似。计算过程是：首先计算索引表达式，得到的索引数组与表达式 a 的结果相似，然后再用索引数组分别处理数组 x 和数组 y。

表达式 f 的索引数组建立条件是 y 数组元素小于 0。但是，y 数组中没有小于 0 的元素，因此索引数组的元素全为 0，结果没有从 x 数组中选出任何元素，表达式 f 的输出是一个空数组。

逻辑索引是一种很有用的方法，可以有效地用于计算复杂的函数关系。本章末尾还有关于逻辑索引的习题。

2.4.5 元胞和元胞数组

前面所讲的 MATLAB 数组都默认数组元素属于同一种数据类型，但是，在某些复杂的应用场合，有时需要将不同类型的数据组合在一起，这时，就需要使用可以容纳不同数据类型的数据结构。

元胞数组的每个元素称为元胞，元胞可以是数字、数组、字符、字符串等任何不同类型的数据，甚至元胞本身也可以是一个元胞数组。在允许范围内，元胞数组可以有任意的大小和形状，例如，可以是多维结构数组。

普通数组的下标用圆括号里的数表示，如 A（1，2），元胞数组的下标则用花括号表示。

例 2.12 元胞数组以及混合数据类型

生物信息学中广泛应用的基因表达数据库中有一个库的数据来自于 Iyer 等人 1999 年发表的论文“人类成纤维细胞的血清应答转录程序”（The transcriptional program in the response of human fibroblasts to serum）。该数据库中的每个基因都有以下几个属性：基因簇序列号、Genbank 编号、克隆 ID、基因名称以及 24h 内表达的变化情况。访问以下网站可以查看该数据库：

http://gepas.bioinfo.cnio.es/data/fibroblasts/fibroblasts_ori.html.

请设计一个数据结构存放该数据库的 517 个基因，并包括以上这些属性。

解：

数据库的整体构架用数组，这样，可以使访问每组基因数据的时间相同。但是，数组中数据的类型是不均一的，既有数值，又有字符串。数值数据有：基因簇序列号、克隆 ID 以及在 0、0.25、0.5、1、2、4、6、8、12、16、20 和 24h 的表达数据。字符串数据有基因名称和 Genbank 编号。

由于基因数组的每个元素都要有这些属性，因此这个数组既不能是纯字符串数组也不能是纯数值数组，显然要用元胞数组，每个元胞包含这 5 项内容。

构建这个元胞数组的方法有两种，两者的区别在于基因表达数据的组织方式不同。一是把所有的表达数据存放在一个数组里，二是直接把每个表达数据作为元胞数组的一个元素来存放。

以下 MATLAB 程序以成纤维细胞数据库为例，运用第一种方法建立数据结构。所用的成纤维细胞数据为

基因簇序列号	4			
Genbank 编号	W88572			
克隆 ID	417426			
基因名称	Home sapiens protein 4.1-G			
在 0、0.25、0.5、1、	1	0.97	1	0.85
2、4、6、8、	0.84	0.72	0.66	0.68
12、16、20、24h 时的表达数据	0.47	0.61	0.59	0.65

整个元胞数组为 517 行乘 5 列，每列对应一个属性，其中有一列本身又是数组，存放基因的表达数据。数组第一行用于存放以上成纤维细胞数据，因此，输入的基因簇序列号为 4，该行的 5 个属性都与这个序列号相关。元胞数组第一行各个元素的赋值指令如下：

```
    Fibro{1,1} =4;
    Fibro{1,2} =417426;
    Fibro{1,3} =[1  0.97  1  0.85  0.84  0.72  0.66  0.68  0.47
0.61  0.59  0.65];
    Fibro{1,4} = 'Homo sapiens protein 4.1 - G';
    Fibro{1,5} = 'W88572';
```

这里使用的是两种元胞数组索引方法之一。输入这些数据之后，MATLAB 工作空间中就有了 Fibro 变量，它既不是字符数组，也不是双精度数的数组，而是元胞数组类型。

数据类型“结构”是 MATLAB 的一种多维数组，其中的元素可以用文本形

式的域名来访问，不用整数下标。上述例题很容易改成使用符号名称访问数据，而不用数值下标。

例 2.13　结构数组以及混合数据类型

不用元胞，而用 MATLAB 的结构数据类型表示上述例题的数据。

解：

以下 MATLAB 程序显示了如何将同样的基因表达数据输入到结构数组而不是元胞数组中。注意，这里要用圆括号表示数组元素，不用元胞数组的花括号。

```
Fibro(1).number =4;
Fibro(1).cloneid=417426;
Fibro(1).expprofile =[1  0.97  1  0.85  0.84  0.72  0.66
0.68  0.47  0.61  0.59  0.65];
Fibro(1).name = 'Homo sapiens protein  4.1 - G';
Fibro(1).accession = 'W88572';
```

其中，域名 cloneid 是基因的克隆 ID，域名 expprofile 是基因的表达数据，域名 accession 是编号。

在 MATLAB 指令行输入这些指令之后，工作空间中就有了 Fibro 变量，该变量是结构数组类型，不是字符、双精度数，也不是元胞数组。

2.4.6　MATLAB 没有明确定义的数据结构

常规指令式编程语言以及面向对象编程语言的算法中常用到许多数据结构，如堆栈、队列、链表和树等，可以方便地组织数据，完成某些特定的操作。数组中的每个元素可以直接访问，而这 4 种数据结构元素的访问具有特定的控制方式，不过，存放这些数据结构对象的内存空间是开放的，存储的大小没有固定的限制。堆栈、队列和链表这 3 种数据结构按照序列存放数据，而树则分层次存放数据。有关这些数据结构的 MATLAB 实现已超出了本书的范围，下面只举例说明。详细内容请参阅本章末尾列出的参考书[1]。

例 2.14　MATLAB 的数据结构——堆栈的实现

堆栈这种数据结构只能在数据的一端存入或取走数据，与数组不同，不能随机访问数据，并且，每次只能访问一个数据，就是“栈顶”的这个数据。MATLAB 函数 pop（ ）和 push（ ）用于访问栈顶，pop（ ）是取走栈顶的数据，push（ ）则向栈顶存入一个新的数据。

[1] 原书第 2 章末尾并没有列出参考书。——译者注

假设有一个字符串表示的氨基酸序列，请说明如何利用堆栈求其反转序列。

解：

本题的程序分两部分：一是建立堆栈的操作函数；二是利用堆栈反转氨基酸序列。

堆栈用 MATLAB 数组表示，保留数组的第一个元素，用于存放堆栈元素的数目。push（ ）和 pop（ ）两个 MATLAB 函数用于完成入栈和出栈操作。push 函数有两个参数：第一个是堆栈名，第二个是放入栈顶的数据。pop 函数有一个参数，即堆栈名；并返回一个数值，就是栈顶的数据。以下 MATLAB 函数 push. m：

```
% push.m
function push(A,x)
t=A(1);
A(t+1)=x;
A(1)=t+1;
end
```

其中，x 是要放入堆栈 A 栈顶的数据。此函数首先读取堆栈中元素的数目（其值刚好等于目前栈顶元素下标值减 1），再将 x 入栈，放在数组下一个空元素的位置。最后，堆栈元素数目增 1。MATLAB 函数 pop. m 如下：

```
% pop.m
function x=pop(A)
t=A(1);
x=A(t+1);
A(1)=t-1;
end
```

此函数首先读取堆栈数据数目和栈顶元素，将栈顶元素存入变量 x，然后把堆栈数据的数目减 1，因为栈顶元素已取走，不再属于堆栈。注意，此函数假设堆栈中至少有一个元素存在。以下 MATLAB 函数 reverse. m 反转氨基酸序列：

```
% reverse.m
function y=reverse(x)
l=length(x);
Stack=[];
Stack(1)=0;
for i=1:l
```

```
        c = x(i);
        push(stack,c);
    end
    for i = 1:l
        y(i) = pop(Stack)
    end
```

此函数首先求序列的长度并初始化空堆栈，然后，将每个代表氨基酸的字符入栈。当所有字符都入栈之后，再将字符出栈，并依次存放到新的字符串中。下面是该函数运行的一个例子：

```
>> reverse('AACTGACT')
ans =
TCAGTCAA
```

建议读者用纸和笔跟着程序的操作顺序走一遍，证实氨基酸序列确实是被反转了。

2.4.7 数据类型转换

与其他模块化结构程序设计语言一样，MATLAB 也具有将一种数据类型转换成为另一种数据类型的功能。表 2.1 列出了几个 MATLAB 指令，可以把行标题指定的数据类型转换为列标题指定的数据类型。

表 2.1 用于数据类型转换的 MATLAB 指令

	字 符 串	元 胞
数	num2str	num2cell
整数	int2str	
矩阵	mat2str	

指令 T = num2str（X）将数 X 转换为字符串 T，大约包括 4 位数字，可能再加上指数。指令 I = int2str（N）则将整数 N 转换为字符串 I。如果输入参数不是整数，则先四舍五入再转换。这两条指令的输入参数都可以是单个数值或矢量（即数值矩阵）。指令 C = num2cell（A）将矩阵 A 转换成元胞数组，把 A 的每个元素放入一个独立的元胞，元胞数组的大小与矩阵 A 相同。最后，指令 M = mat2str（A）将二维矩阵转换成一个字符串 M，而指令 eval（M）又可以将字符串重新变成原矩阵 A。

num2str 和 int2str 之类的函数在作图显示包含数值的标题时很有用。

例 2.15 数据类型的转换

调节函数 $e^{-\theta}\sin^2\theta$（例 2.3 所示）的参数，使其曲线尽可能接近心电图

（ECG）的复合波，作出该曲线图，并在图上标出每次测试所用的参数值。

解：

例 2.3 中的函数曲线 $e^{-\theta}\sin^2\theta$ 很像 ECG 信号的 QRS 复合波和 T 波。你可以改变 $e^{-\alpha\theta}\sin^2\beta\theta$ 式中系数 α 和 β 的值，作出波形图并用一个标题显示 α 和 β 的值，找到模拟 QRST 复合波的最佳系数组合。

要显示参数值，就要用到 num2str 函数，将 α 和 β 的值转化成字符串，显示在图形的标题中。标题用 title（ ）函数显示，此函数带一个字符串类型的参数。下面编写的 findecg 函数输入两个参数（即 α 值和 β 值），计算 $e^{-\alpha\theta}\sin^2\beta\theta$ 的函数值，作出曲线，并在曲线图上显示函数式作为标题，以便找出最接近 QRST 波形的数学模型。

```
% Example 2.15
function findecg(a,b)
twopi =2 * pi;
twotheta =0:(twopi/100):twopi;
plot(twotheta,exp(-a * twotheta).* (sin(b * twotheta)).^2);
title(['Plot of e^{-',num2str(a),'\theta} * sin^2 ',...
    num2str(b),'\theta']);
```

以下是 findecg 函数运行的一个示例：

```
>> findecg(3,3)
```

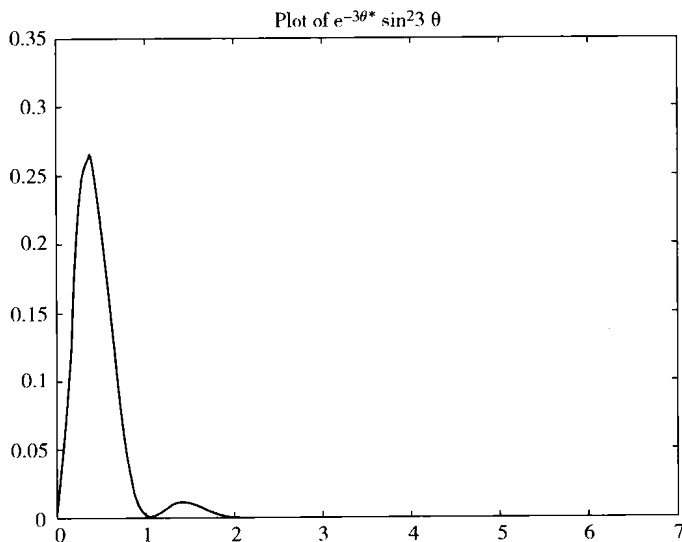


图 2.4 函数 num2str 的用法

注意，图2.4的标题包含了传递给findecg函数的两个变量值。标题用title函数显示，title（ ）只有一个参数，本例中就是几个字符串形成的一个矢量。文本字符串都放在单引号里，有两处调用了num2str函数，把参数值转化为字符串。字符串中的“^”符号用于形成上标，特殊字符串“\theta”则用于产生“θ”字符。

2.5 面向对象系统简介

在过去的20年里，面向对象程序设计（Object-Oriented Programming，OOP）的概念引起了计算机程序设计的巨大变革。OOP程序不是围绕操作而是围绕对象组织的；也可以看成OOP是围绕数据组织的，而不是按照逻辑组织的。了解OOP的原理，有助于正确编写和调试MATLAB程序，避免使用不正确的运算符和表达式。

传统上把计算机程序看成一个逻辑过程，即接收输入数据、处理数据，然后产生输出结果。因此，编程的关键是写出逻辑过程，而不是定义数据。常用的结构式程序设计方法较好地制定了一套简明的规则，有助于编程人员组织程序的逻辑过程。

OOP语言就不同了。编程人员注重数据，也就是要操作的对象，而不是操作数据所需的逻辑过程。操作对象的范围很广，可以是人（用姓名、住址等属性描述），可以是建筑物及其楼层（其属性可以设置），还可以是计算机桌面上的小部件，如按键和滚动条等。

对象是一种软件元素，里面包含相关数据（即变量）以及处理这些数据的方法（即过程子程序）。对象是类的实例，它通过初始化来创建，也就是建立一个属于某个特定类的对象实例。例如，“加利福尼亚”是“州”这个类的一个实例。

方法是对象中包含的过程子程序，其他对象要求这个对象服务时也可以调用这些方法。

消息是在对象之间传递的信息，接收到消息的对象就要执行某个方法来响应这个消息。消息由3部分组成：接收对象名、要执行的方法名以及该方法运行时所需的参数。

类是一个模板，用于定义某种对象的变量和方法。某个类的所有对象具有相同的形式和行为，只是变量中包含的数据不同。

封装是把数据及其相应的过程子程序组合在一起的方法。在OOP语言和系统中，对象就是封装的结果。

一组类之间的关系用树结构表达，这称为类的继承。父类是处于较高层次的

类，通用性较强；子类则是其父类的一种特例。

类方法是一种特殊的方法，由发送给类的消息直接调用，不需要类的实例。类方法通常用于完成某些类的实例不能也不应该做的事，例如，创建和消除类的实例等。

类变量是一种特殊的变量，变量值存放在类的定义中，而不是在类的实例中。类变量的数据对类的所有实例都是一样的。

通过继承这种机制，类可以使用排在高层次的那些父类中定义的方法和变量。

实例方法就是常规的方法，这种方法通过给实例（而不是类）发送消息来调用。“实例方法”这个名称是为了区分常规方法和较少使用的类方法而提出的。

实例变量就是常规的变量，用于存放类的各个实例的不同数据。用这个名称也是为了区分常规变量和较少使用的类变量。

例如，“哺乳动物”这个类可以有“人”、“猫”和“狗”这些子类，“狗”又可以有“猎兔犬”、“杜宾犬”和“牧羊犬”这些子类。“哺乳动物”是“人”的父类。“猎兔犬”这个实例继承其父类“狗”具有的操作（包括消息和方法）。

Smalltalk 是最早出现的面向对象的计算机语言之一，C++和 Java 是如今最流行的 OOP 语言，Java 专门用于分布式系统的设计。MATLAB 也是一种 OOP 语言。下面用一个简单的数学运算说明 OOP 的基本原理。

假设你编写了计算机程序语句

$$z = x + y$$

如果用的是传统模块化语言，则该语句会被编译成以下几步：

1）读取变量 x 的值，转化为双精度数，并存放在寄存器 R1 中；

2）读取变量 y 的值，转化为双精度数，并存放在寄存器 R2 中；

3）将寄存器 R1 和 R2 的数相加，把双精度数的和转化为变量 z 的类型。

如果用的是 OOP 语言，则变量 x 是一个对象，传送的消息是“+”，参数是变量 y，要执行对应于消息“+”的方法（也就是加运算）。关键是具体执行哪个方法由对象 x 的类型决定。如果 x 是整数，就执行整数加和方法；如果 x 是双精度数，则执行双精度数加和方法。

注意，调用不同的方法时可以用同样的消息符号（即消息名），这被称为运算符重载，在 MATLAB 等 OOP 系统中很常见。下面这个例题说明了 MATLAB 运算符的重载功能。

例 2.16 顺序语句组成的面向对象的简单程序

设 $x = [2\ 5\ 1\ 6]$，计算如下各个运算的结果：

1）数组的每个元素加 16；

2）下标为奇数的元素加 3；

3）求每个元素的平方根；

4）求每个元素的平方。

解：

这 4 个运算的结果与 MATLAB 的面向对象的性质有关。在 MATLAB 里，可以把两个数相加，也可以把数与矢量相加，要根据 MATLAB 操作的定义给出结果。

1）对象 $\boldsymbol{x}$ 是一个整数矢量，传送给对象的消息是“加”，消息的参数是一个整数对象。响应“给一个矢量加上一个对象”这一消息时，被调用的方法首先检查这种类型的参数是否可以加到一个矢量上。此处，参数是整数类型，将一个整数与一个矢量相加，MATLAB 的方法有明确的定义，就是指将整数与每个元素相加。

编程时，首先创建矢量并赋给一个变量，然后将矢量加上常数 16。MATLAB 程序为

```
x = [2 5 1 6]
x + 16
```

输出结果为

```
x =
      2    5    1    6
ans =
18    21    17    22
```

2）由于这个矢量只有 4 个元素，因此，要把奇数下标的元素加 3，最简单的方法就是写一个新的表达式，更新矢量第 1 和第 3 个元素的值，即

```
[2 +3 5 1 +3 6]
```

输出结果为

```
ans =
    5    5    4    6
```

其实，如果是同样的运算，不必像这样把矢量每个元素都写出来，也可以用逻辑数组索引的方式完成（见第 2.4.4 节）。

3）这里求的是矢量每个元素的平方根。MATLAB 是面向对象的，因此，只需编程求矢量的平方根即可

```
sqrt(x)
```

输出结果为

```
ans =
    1.4142    2.2361    1.0000    2.4495
```

其中，对象是矢量 x，消息是 MATLAB 函数 sqrt（ ），被调用的方法检查这个矢量是否可以计算平方根，结论是可以计算，并得到以上输出结果。

4）常规的 OOP 运算中没有对矢量元素求平方的运算。MATLAB 指令 x^2 传送的消息是求对象 x 的幂“^”，参数为 2。如果消息“^”传给一个矩阵（x 是 1 ×4矩阵），意思就是执行矩阵自己乘自己的运算。于是，被调用的方法首先检查这个矩阵是不是方阵。这里的 x 是 1 ×4 矩阵，不是方阵，因此指令就会产生一个错误信息，不会输出元素的平方值。

所以，这里给定的消息必须是指对象每个元素的平方。MATLAB 的元素乘元素消息用“.^”符号表示，如

```
x.^2
```

输出结果为

```
ans =
  4    25    1    36
```

2.6 算法分析和程序分析

编写 MATLAB 程序的正确方法不是惟一的。作为工程师，你可以选择很多种表达数据和实现算法的方法。如果要处理的数据量较小，则各种程序的计算时间以及内存的使用量之间的差别不会很大。但是，当数据量很大时，比如图像重建、长时间的多通道脑电图或心电图记录等，如何选择算法就变得很重要。

生物医学工程师如何选择算法呢？你需要考虑如下几个因素：

1）结果的准确度和精度；

2）程序运算所需的时间；

3）程序运行所需的内存空间。

本节将介绍一些分析工具，有助于决定如何表达数据以及如何选择算法的实现方法。

2.6.1 算法的复杂度

确定某种计算所需要的时间以及内存空间的过程，称为算法分析。测定计算机程序的实际运行时间并不是衡量时间的好方法，因为随着处理器芯片的新发展，计算机的运算速度在不断提高。

如果事先不知道有多少数据量，或者要编写一个数据量适用范围较大的通用程序，就需要进行算法分析，以便提供选择算法的依据。

实际运行时间（即时钟时间）有多个影响因素，包括处理器时钟的速度、内存的大小以及是否有其他任务在同时运行等。通常所说的程序运行速度，惟一的标准化指标就是渐近运行时间，也称为复杂度。该时间指标是数据量的函数。

2.6.2 运算时间的计算

确定算法的渐近时间复杂度有几条通用的规则，首先，假设基本时间单位是单个加法运算所需的时间，例如

$$Z = x + y$$

此处的赋值运算时间小于加法运算，因此，假设赋值时间等于加法运算时间，这条指令运行的总时间就是 2 个加法运算时间。而以下乘法的运算时间要长：

$$C = a * b$$

乘法的时间近似等于加法时间乘以数的长度。

通常，估算运行时间的基本测量单位，要么用加法时间，要么用乘法时间，取决于被求解的问题。哪种运算用得较多，就用哪种运算作为基本测量单位。

计算语句序列组成的算法的运行时间时，有以下规则：

规则 1：两个数求加法或求减法所需的时间为 1 个单位时间，两个数求乘法或除法所需的时间为数的长度（即数的二进制位数）乘以加法的时间。

因为乘法是用重复的加法完成的，如果单个加法的时间一定，设为 1，那么乘法所需的时间就是加法运算的次数，也就是二进制数的位数。

规则 2：语句序列的运行时间就是各条语句所需时间之和。

$$X = 10 + Y$$
$$Z = X * 3$$
$$Y = Z/X$$

计算 X、Y 和 Z 所需的总时间就是各条语句运行时间之和。记住，乘法运算所需的时间是数据长度乘以加法运算所需的时间，因此，计算 Z 和 Y 的时间要远大

于计算 X 的时间。计算这 3 个值所需的总时间主要由乘法和除法决定，相比之下，计算 X 的时间可以忽略不计。

由于计算一个加法的时间是常数，因此，计算一个乘法（或除法）的时间也是常数，计算 X、Y 和 Z 的总时间也一定，可以表示为运算的次数，于是，总时间就是语句序列长度的函数。

规则 3：循环语句的运行时间为循环体的运行时间乘以循环次数。
如下单循环：

```
for j =1 to n
        sum = sum + a(i) * b(j)
end
```

设 w 为计算机的字长，则循环体的运行时间将是 $w+2$ 数量级，其中 w 为乘法时间，另外 2 个单位时间是加法时间和赋值时间。这条语句要重复 n 次，所以循环执行的总时间为 $n*(w+2)$，也就是 $nw+2n$。字长 w 肯定大于 2，nw 这项的增长速度要比 $2n$ 大得多，因此，可以确定运行时间为 nw 数量级，记为 $O(nw)$。

如果程序中有嵌套循环，则要把每个循环的重复次数相乘，例如

```
for i =1 to m
    for j =1 to n
        sum = sum + a(i) * b(j)
    end
end
```

其中内循环体有 $w+2$ 步运算，内循环共有 $n*(w+2)$ 步运算，整个嵌套循环则有 $m*n*(w+2)$ 步运算。当 m、n 和 w 增加时，计算式中 $2mn$ 项比 mnw 项的增长速度小得多。因此，只需保留计算式的最大项，渐近运行时间记为 $O(mnw)$。

例 2.17 分析程序运行时间与数据量大小的关系

请编写一个程序，计算单通道 ECG 时间序列的能量。并用 3 组 ECG 采样数据测试这个 MATLAB 程序：第一组有 128 个采样点，第二组有 4096 个采样点，第三组有 65536 个采样点。

能量可以用两种方法计算，一是计算电压幅值平方的总和，二是计算频率分量幅值平方的总和。第二种方法可以分别用 DFT 或者 FFT 算法求频谱，因此又可以分成两种算法。

随着时间序列采样点数的增加，程序运行时间也会增加，请确定哪种算法的运行时间最短。

解：

由题目可知，计算 ECG 信号能量的方法共有如下 3 种：

1）计算电压幅值平方的总和；

2）用 DFT 求 ECG 信号的频谱，再计算频率分量幅值平方的总和；

3）用 FFT 求 ECG 信号的频谱，再计算频率分量幅值平方的总和。

根据帕塞伐尔（Parseval）定理，这 3 种算法在理论上是等价的。也就是，时域上信号的能量等于频域上信号的能量。除了运算结果的准确度以外，这 3 种算法的区别就是所需的计算时间不同。

方法 1：设 ECG 数据存放在 MATLAB 数组 ecg 中，计算能量的程序为

```
power = 0;
for j = 1 to length(ecg)
    power = power + ecg(i) * ecg(i)
end
```

如果 ECG 数组的长度为 n，并且把一次乘法运算作为基本时间单位，那么，根据上述规则 3，该算法的运行时间为 $O(n)$。

方法 2：如果数组长度不是 2 的幂，MATLAB 的 fft 函数就是计算 DFT，程序如下：

```
freq = fft(ecg)
power = 0;
for j = 1 to length(freq)
        power = power + freq(i) * freq(i)
end
```

根据上述规则 2，这种方法的运行时间等于 DFT 计算时间加上能量计算时间，从 MATLAB 的帮助信息中可知，DFT 运算的时间为 $O(n^2)$，其中 n 为 ECG 数组的长度。则整个程序的运行时间为 $O(n^2+n)$。由于 n^2 比 n 的增长速度要快得多，因此，可以忽略第二项，运行时间就是 $O(n^2)$。

方法 3：为了在 MATLAB 中计算 FFT，数组的长度必须是 2 的幂，其计算时间为 $O(n\log_2 n)$。程序总的运行时间为 $O(n\log_2 n+n)$，这里也可以忽略第二项，运行时间就是 $O(n\log_2 n)$。因为 $\log_2 n$ 小于 n，FFT 的计算时间小于 DFT 的计算时间。

由此可见，运行时间最少的算法是方法 1，它直接从时间序列计算能量。图 2.5 显示了 $O(n)$ 、$O(n\ \log_2 n)$ 和 $O(n^2)$ 这 3 个复杂度之间的大小区别。

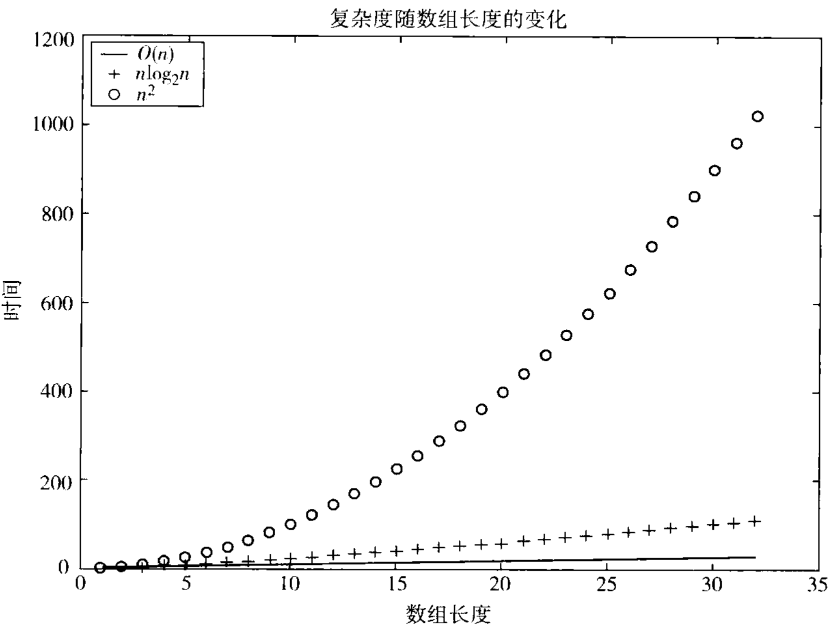


图 2.5 $O(n)$、$O(n\log_2 n)$ 和 $O(n^2)$ 复杂度大小区别

2.7 本章学习要点

学完本章之后，读者应该掌握以下内容：

1）计算机应用于生物医学工程，可以不出差错地重复执行任务，并且提供友好的人机交互界面。

2）模块化的程序设计语言有几种控制结构，即顺序语句、条件执行和重复循环。每种结构的控制流都只有一个入口和一个出口。规定使用少数几种控制结构，可以使编程人员易于开发和调试程序。

3）在传统的模块化结构语言中，程序由一系列操作组成。

4）在面向对象程序设计语言中，程序是数据对象及其相互关系的说明。MATLAB 是一种面向对象程序设计语言，可以方便地实现矩阵运算。

5）数据类型和数据结构是表达和组织数据集合的方法。数值和字符串是 MATLAB 的简单数据类型。数组用于表示同类型的数据集合，元胞数组和结构数组则可以表示混合类型的数据集合。

6）算法及其实现过程的简单分析可以帮助编程人员估计程序运行时间随数

据量增加的变化情况。程序所用的内存量与计算时间，这两者之间始终需要折中。生物医学领域的工程人员要根据这些信息来决定如何编写程序。

7）如何用 MATLAB 实现简单的计算。

8）如何设计和编写 MATLAB 程序，包括脚本程序和函数。

2.8 习题

2.1 求以下表达式的值，先手工计算，然后再用 MATLAB 验证答案。

```
a. 2/2 * 3
b. 6 -2/5 +7^2 -1
c. 10/2\5 -3 +2 * 4
d. 3^2/4
e. 3^2^2
f. 2 + round(6/9 +3 * 2)/2 -3
g. 2 + floor(6/9 +3 * 2)/2 -3
h. 2 + ceil(6/9 +3 * 2)/2 -3
```

2.2 建立一个矢量 $\boldsymbol{x}$，其元素值为 $x_n = (-1)^{n+1}/(2n-1)$，共有 100 个元素，求所有元素之和。

2.3 以下表达式用于模拟美国人口增长的速度，请作出表达式随时间 t 变化的曲线

$$P(t) = \frac{197\ 273\ 000}{(1 + e^{-0.0313(t-1913.25)})}$$

其中，日期 t 的单位是公元年，取 $t = 1700 \sim 2000$，并估计 2020 年的人口为多少。

2.4 设数组 $A = [2\ 7\ 9\ 7;\ 3\ 1\ 5\ 6;\ 8\ 1\ 2\ 5]$，请解释以下指令的运行结果：

```
a. A'
b. A(:,[1 4])
c. A([2 3],[3 1])
d. reshape(A,2,6)
e. A(:)
f. flipud(A)
g. fliplr(A)
h. [A A(end,:)]
i. A(1:3,:)
j. [A;A(1:2,:)]
k. sum(A)
l. sum(A')
m. sum(A,2)
```

n. `[[A;sum(A)][sum(A,2);sum(A(:))]]`

2.5 设 $x=$ [3 15 9 12 -1 0 -12 9 6 1]，请写出完成下列操作的指令：

a. 将 x 中的正数设置为 0；

b. 将其中 3 的倍数设置为 3（这里可以用 rem 函数）；

c. 将 x 中的偶数乘以 5；

d. 提取 x 中大于 10 的数，存入另一个 y 变量；

e. 将 x 中小于平均值的数设置为 0；

f. 将 x 中大于平均值的数设置为该值与平均值之差。

2.6 请编写一个 MATLAB 函数，求以下函数的值：

$$t(y)=\begin{cases}200 & \text{当 } y \text{ 小于 } 10000 \text{ 时}\\ 200+0.1(y-10000) & \text{当 } y \text{ 位于 } 10000 \text{ 与 } 20000 \text{ 之间时}\\ 1200+0.15(y-20000) & \text{当 } y \text{ 位于 } 20000 \text{ 与 } 50000 \text{ 之间时}\\ 5700+0.25(y-50000) & \text{当 } y \text{ 大于 } 50000 \text{ 时}\end{cases}$$

测试数据如下：

a. $y=50000$ $t=200$

b. $y=17000$ $t=900$

c. $y=25000$ $t=1950$

d. $y=75000$ $t=11950$

如果你编写的脚本用 input 输入 y 参数，就可以用这些测试数据验证程序的正确性。请解释为什么以下 if 模块不能给出本题的正确答案。

```
if y<10000
t=200
elseif 10000<y<20000
t=200+0.1*(y-10000)
elseif 20000<y<50000
t=1200+0.15*(y-20000)
elseif y>50000
t=5700+0.25*(y-50000)
end
```

2.7 设 $x=[8\ 2\ 2]$，$y=[7\ 9\ 3]$，求以下数组的值：

a. $a_{ij}=x_iy_j$

b. $b_{ij}=x_i/y_j$

c. $c_i=x_iy_i$ 然后求 c 的元素之和

d. $d_{ij}=x_i/(2+x_i+y_j)$

e. $e_{ij}=x_i$ 和 y_j 中较小的数的倒数

2.8 勒让德多项式（Legendre polynomial）$P_n(x)$ 用以下递推方程定义：

$$(n+1)P_{n+1}(x)-(2n+1)P_n(x)+nP_{n-1}(x)=0$$

设 $P_0(x)=1$，$P_1(x)=x$，$P_2(x)=(3x^2-1)/2$，求后续3个勒让德多项式，并在 $[-1, 1]$ 区间上作出这6个函数的曲线。

2.9 请编写一个函数，计算矢量中元素的累积乘积。矢量 x 中第 j 个元素的累积乘积 x_j 定义为

$$P_j=(x_1)(x_2)\cdots(x_j)$$

其中，$j=1:x$ 的长度。用以下两种不同的方法分别编写这个函数：

a. 用双重 for 循环，一个元素一个元素地求值。内循环求累积乘积，外循环扫描整个 P 矢量的所有元素。

b. 用 MATLAB 内置函数 prod 替代 a 中的内循环。

可以用 MATLAB 内置函数 cumprod 验证这两个程序的正确性。

2.10 在例2.3 和例2.15 所作的函数曲线中，当参数 m = 's'，$n=2$ 时，函数曲线很接近 QRS 波加 T 波的复合波形状。请编写 MATLAB 脚本，产生包含3个这种复合波的时间序列。假设心率恒定。

第3章　数值分析的概念

3.1　科学计算

科学计算研究科学技术各个领域所产生的数学问题的求解算法。生物医学领域的工程人员首先需要用连续数学模型解释所观察到的生物现象和化学现象，本书的例题表明，要求解这些模型的解析解通常很困难，甚至是不可能的。

数值分析是有关利用计算机求解数学模型近似解的算法的数学理论。用计算机求解时，先用有限的数值数组来近似连续数学函数，再利用算法有效、准确、可靠地近似求解数学问题。

设计数值算法时，需要考虑很多因素。本书的目的就是使生物医学领域的工程人员掌握必要的方法，并且举例说明如何在实际工作中应用数值分析和数值方法。如果需要了解有关的背景知识，读者可以参阅本章末尾的参考文献。
本章包括如下学习内容：

- 绝对误差和相对误差之间的区别；
- 如何利用泰勒级数（Taylor series），即泰勒公式，展开连续函数；
- MATLAB 中如何用浮点数表示实数；
- 数值计算过程中舍入误差是如何传播的；
- 避免消去误差的几种方法。

3.2　数值算法及其误差

用计算机近似处理数学模型时，计算中不可避免地总是存在误差，计算结果只能是近似的，只能尽可能减小误差。生物医学领域的工程人员在设计数学模型数值解法的时候，要了解相对误差和绝对误差等可能的误差来源，并懂得如何减小这些误差。

给定某个量 u 及其近似值 v，则 v 的绝对误差为 $|u-v|$，相对误差为 $|u-v|/|u|$（设 $u\neq0$）。通常相对误差比绝对误差更有意义，特别是对于浮点数表示法，3.3 节会说明这一点。表 3.1 比较了几种 u 和 v 具体数值组合的绝对误差和相对误差。

表 3.1　绝对误差与相对误差的比较

u	v	绝对误差	相对误差
1	0.99	0.01	0.01
1	1.01	0.01	0.01
-1.5	-1.2	0.3	0.2
100	99.99	0.01	0.0001
100	99	1	0.01

当 $u \approx 1$ 时，绝对误差与相对误差之间的区别不大；但是，当 $|u| \gg 1$ 时，相对误差就能很好地反映 u 与近似值 v 之间的差别。

可能影响数值计算准确度的误差有如下几种。建立数学模型时，会引入模型的近似误差。例如，计算星体的特性时，经常把它们看作球体；为了使复杂的化学反应数学模型易于处理，经常忽略反应的某些不重要部分。计算中所用的一些初始输入数据通常由实际测量得到，由于测量系统和仪器不可能绝对准确，即使经过计算机模型的仔细计算，得到的结果也不会与真实值一模一样，这就产生了测量误差。第 9 章会讲解这个问题。

在确定待求解问题的准确性时，需要考虑模型误差和测量误差等误差因素。

计算过程中用近似公式代替实际函数会产生近似误差。近似误差有截断误差和收敛误差两种。连续过程的离散化采样时，会产生截断误差，例如，插值近似、数值微分和数值积分等。迭代方法会产生收敛误差，例如，非线性问题通常必须用迭代法近似求解，经过无限次迭代之后，其极限值会收敛到准确解；但是，实际处理时只进行有限次迭代，就会产生误差。迭代法也经常用于线性代数，经过有限次迭代，还未达到准确解时，迭代过程就结束了，因此，就产生收敛误差。第 4 章将讲述这些问题。

即使没有近似误差，在非整数的实数计算中也都会有舍入误差，这是因为计算机无法完全准确地表示实数。本章 3.5 节将讨论 MATLAB 的实数表示法。在计算过程中，舍入误差总是向前传播，其大小甚至可能超过算法的模型误差和近似误差。由此可见，在选用特定的数值方法时，要认识其近似误差和舍入误差，并尽可能控制误差。

3.3　泰勒级数

连续形式的方程通过泰勒级数（以及泰勒公式）可以转化为离散形式的方程，因此，泰勒级数是连续方程与离散方程之间相互联系的关键。设 $f(x)$ 在包

含 x_0 和 x_0+h 的某个区间上存在 $k+1$ 阶导数，则其无限泰勒级数为

$$f(x)=f(x_0)+(x-x_0)f'(x_0)+\frac{(x-x_0)^2}{2}f''(x_0)+\cdots+\frac{(x-x_0)^k}{(k)!}f^{(k)}(x_0)+\cdots \tag{3.1}$$

有限泰勒级数为

$$f(x)=f(x_0)+(x-x_0)f'(x_0)+\frac{(x-x_0)^2}{2}f''(x_0)+\cdots+\frac{(x-x_0)^k}{k!}f^{(k)}(x_0)+\frac{(x-x_0)^{k+1}}{(k+1)!}f^{(k+1)}(\xi) \tag{3.2}$$

这两者是等价的。有限泰勒级数最后一项

$$R^{n+1}=\frac{(x-x_0)^{n+1}}{(n+1)!}f^{n+1}(\xi) \tag{3.3}$$

称为余项，ξ 为 x_0 与 x 之间的某个点。

用式（3.1）计算与 x_0 点相距 h 的点 x_0+h 上的 $f(\ )$ 函数值，如果截去式中第 k 项之后的项，就成为泰勒公式，泰勒公式具有误差项，即

$$\frac{h^{k+1}}{(k+1)!}f^{(k+1)}(\xi) \tag{3.4}$$

这就是用有限项泰勒级数近似连续函数所产生的截断误差。

例 3.1 截断误差和舍入误差是如何产生的

求函数 $f(x)=\sin(x)$ 的导数 $f'(x)$ 在 x_0 处的近似值。假设导数 $f'(x_0)$ 不能直接求得，而是需要利用 x_0 附近 x 点的函数值 $f(x)$ 来计算。例如，设 $x_0=0.5$，则有 $f(x_0)=\sin(0.5)\approx 0.479$，请计算 x_0 附近某点 x 的函数值 $f(x)$，并由此计算 $f'(x_0)$。

解：

利用泰勒级数和泰勒公式可以建立算法。对于某个很小的正数 h，有

$$f(x_0+h)=f(x_0)+hf'(x_0)+\frac{h^2}{2}f''(x_0)+\frac{h^3}{6}f^{(3)}(x_0)+\frac{h^4}{24}f^{(4)}(\xi)+\cdots$$

于是

$$f'(x_0)=\frac{f(x_0+h)-f(x_0)}{h}-\left(\frac{h}{2}f''(x_0)+\frac{h^2}{6}f^{(3)}(x_0)+\frac{h^3}{24}f^{(4)}(\xi)+\cdots\right)$$

去掉高阶导数项，其泰勒公式为

$$f'(x_0)\approx\frac{f(x_0+h)-f(x_0)}{h}$$

该算法的截断误差为

$$\left|f'(x_0)-\frac{f(x_0+h)-f(x_0)}{h}\right|=\left|\left(\frac{h}{2}f''(x_0)+\frac{h^2}{6}f^{(3)}(x_0)+\frac{h^3}{24}f^{(4)}(\xi)+\cdots\right)\right|$$

图 3.1 显示了该算法的含义，就是用$f(\ )$邻近点弦线的斜率来近似 x_0 处切线的斜率。

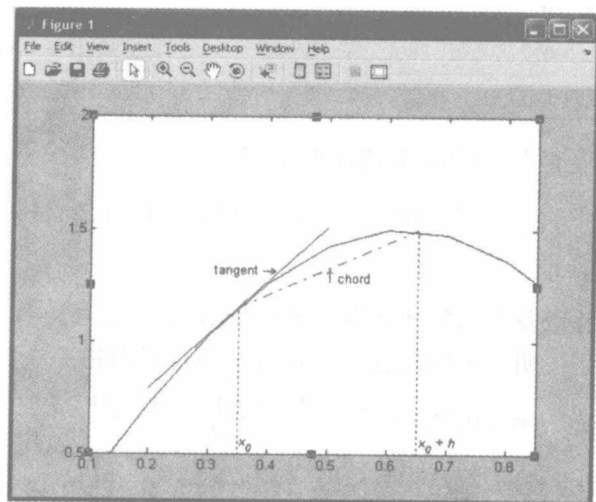


图 3.1　数值方法求导示意图

图中切线（tangent）的斜率$f'(x_0)$用弦线（chord）的斜率$(f(x_0+h)-f(x_0))/h$来近似。

如果$f''(x_0)$已知，并且不等于0，那么，对于很小的h，截断误差可以估计为

$$\left|f'(x_0)-\frac{f(x_0+h)-f(x_0)}{h}\right|\approx\frac{h}{2}|f''(x_0)|$$

即使$f''(x_0)$未知，只要$f''(x_0)\neq 0$，该公式表明，减小h，就可以减小截断误差，截断误差与h成正比。

对于函数$f(x)=\sin(x)$，在$x_0=0.5$处，导数真实值$f'(x_0)=\cos(0.5)=0.877\cdots$。如果取$h=0.1$，用泰勒公式求得的近似值为

$$f'(x_0)\approx(\sin(0.6)-\sin(0.5))/0.1=0.852\cdots$$

绝对误差约为0.025，相对误差$|0.877-0.852|/|0.877|$的值也差不多是这个数量级。

显然，用$h=0.1$计算的$f'(x_0)$不太准确，若采用更小的h值，用同样的泰勒公式求$f'(x_0)$近似值，则误差如下：

h	绝对截断误差
0.1	2.541321e-002
0.01	2.411734e-003
0.001	2.398590e-004
0.0001	2.397274e-005
0.00001	2.397147e-006

可见，误差随着 h 值的减小按比例下降。已知 $f''(x) = -\sin(x)$，并且在 $x_0 = 0.5$ 处，有 $\frac{1}{2}f''(x_0) \approx 0.240$，这个误差列表证明了 $0.24h\left(即\frac{h}{2}|f''(x_0)|\right)$ 是很好的截断误差估计。

注：以上计算以及下面的计算都用 MATLAB 完成。

这些结果似乎说明，采用任意小的 h 值几乎可以求得完全准确的解。例如，假设要使截断误差 $\left|\cos(0.5) - \frac{\sin(0.5+h) - \sin(0.5)}{h}\right| < 10^{-10}$，由例 3.1 的结果可知，似乎只要使算法中 $h < 10^{-10}/0.25$ 即可。但是，对于非常小的正数 h，求得的绝对截断误差值会有如下变化：

h	绝对截断误差
1.0e-8	4.361050e-10
1.0e-9	5.594726e-8
1.0e-10	1.669696e-7
1.0e-11	7.938531e-6
1.0e-13	6.851746e-4
1.0e-15	8.173146e-2
1.0e-16	3.623578e-1

图 3.2 的双对数坐标图显示了误差随步长 h 的变化情况。当 h 值从右到左逐渐减小时，开始阶段误差也逐渐减小，但随后的趋势发生了变化，变为误差逐渐增加。图 3.2 由 MATLAB 脚本产生，程序如下：

```
x0 = .5;
f0 = sin(x0);
fp = cos(x0);
i = -20:0.5:0;
h = 10.^i;
err = abs (fp - (sin(x0 + h) - f0)./h );
```

```
d_err=f0/2*h;
loglog (h,err,'-*');
hold on
loglog (h,d_err,'-.');
xlabel('Step size h')
ylabel('Absolute error')
```

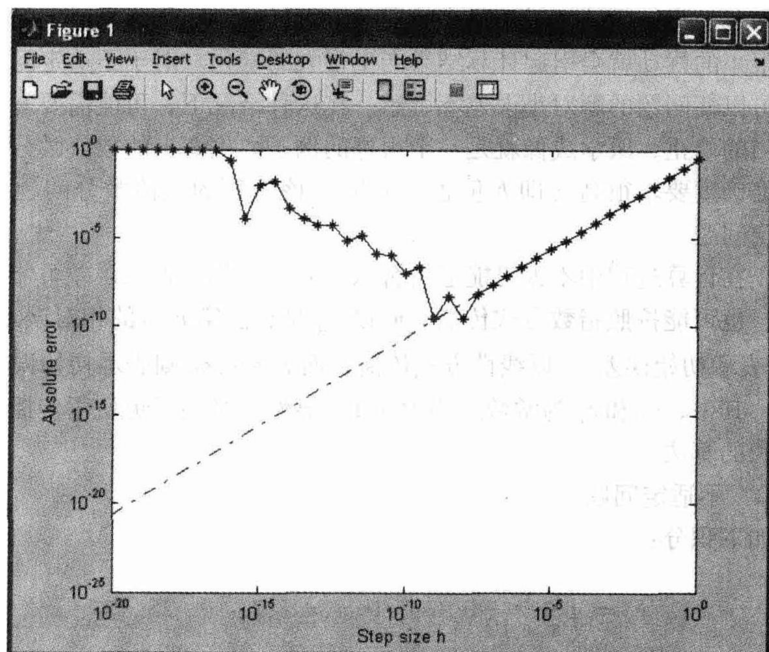


图 3.2　截断误差和舍入误差共同作用的结果

图中实线为函数$f(x)=\sin(x)$在$x_0=0.5$处的$\left|f'(x_0)-\dfrac{f(x_0+h)-f(x_0)}{h}\right|$计算值的插值连线。点划线表示没有舍入误差时的离散化误差变化情况，它是一条直线。

图 3.2 中误差的最小值大约在$h=10^{-8}$处，此处截断误差和舍入误差在总误差中各自占有的比例相当。当h值大于10^{-8}时，截断误差为主；随着h值的减小，截断误差也减小；当h值小于10^{-8}时，截断误差变得很小，舍入误差开始成为误差的主要成分，并随着h值的减小而增大。这就是为什么在涉及求微分方程近似解等数值问题时，要让截断误差为主的重要原因。有关这个重要论题的详细内容请参阅第 7 章。

3.4 减小误差

由于问题求解和数值算法这两步都会有误差，因此计算求得的每个解都存在误差。那么，对于相似的数据和算法，数值解与真实解之间是只有微小的差别呢，还是会有很大的差别？如果差别太大，数值解就毫无意义了。

这个问题在医学成像中很重要，形成了重要的理论，这里只提一下有关的概念，仅供参考。当问题的解对于很小的误差也很敏感时，称为不适定问题，意思是数据中很小的扰动就会引起计算结果产生很大的变化。对于不适定问题，没有任何算法可以使问题的解对小误差不敏感。在这种情况下，可能需要对问题的定义进行适当的修正，医学成像就是一个很好的例子。例 3.1 的数值微分也是一个例子，当准确度要求很高（即 h 值很小）时，该例题的数值微分问题显然就变成了不适定问题。

通常，在计算过程中不太可能避免舍入误差的累积，舍入误差可能按照线性方式传播，也可能按照指数方式传播。假设 ε_n 是算法第 n 步的相对误差，那么，$\varepsilon_n \approx c_0 n \varepsilon_0$ 表示初始误差 ε_0 以线性方式传播；而 $\varepsilon_n \approx c_1^n \varepsilon_0$ 则表示初始误差以指数方式传播。其中，c_0 和 c_1 为常数，均大于 1。显然，必须避免使用传播误差以指数方式增长的算法。

例 3.2 不适定问题

计算如下积分：

$$y_n = \int_0^1 \frac{x^n}{x + 10} \mathrm{d}x \quad 其中, n = 1, 2, \cdots, 25。$$

解：

由微积分可得

$$y_n + 10 y_{n-1} = \int_0^1 \frac{x^n + 10x^{n-1}}{x + 10} \mathrm{d}x = \int_0^1 x^{n-1} \mathrm{d}x = \frac{1}{n}$$

并且

$$y_0 = \int_0^1 \frac{1}{x + 10} \mathrm{d}x = \ln(11) - \ln(10)$$

因此，计算这个积分的数值算法设计为

先求 $y_0 = \ln(11) - \ln(10)$

然后利用下式计算 $n = 1, 2, \cdots, 25$ 时的 y_n 值：

$$y_n = \frac{1}{n} - 10 y_{n-1}$$

图 3.3 是 MATLAB 程序显示的 25 个 y_n 值的序列。如果没有 MATLAB 的浮点数舍入误差，计算值会很准确。但是，实际上每计算一步舍入误差都被乘以 10，因此误差以指数形式上升。显然，对于任何 n 值，y_n 的真实值总是满足 $0 < y_n < 1$，但图中所示的数据序列并非如此，所以，该数值解的结果是错误的。

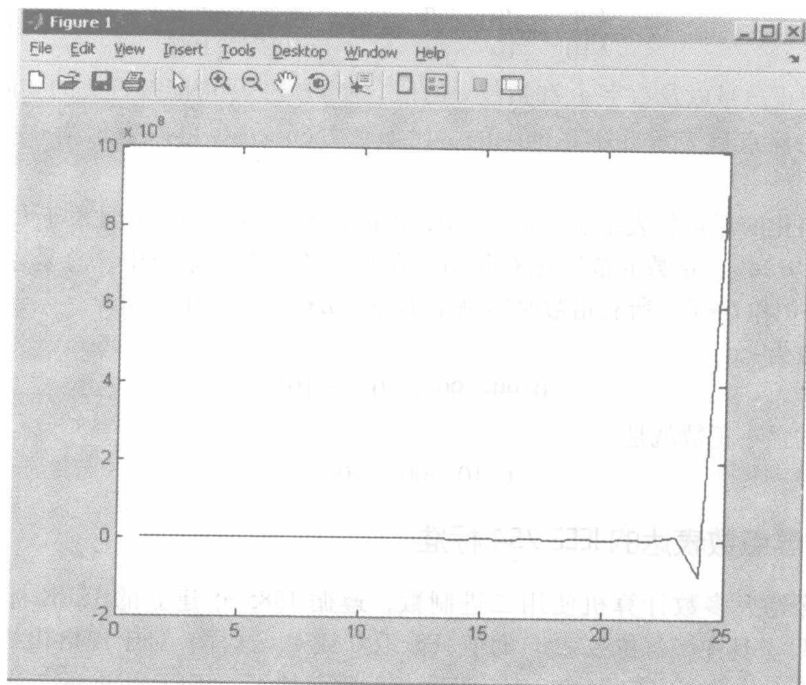


图 3.3 例 3.2 的 25 个数值解数据序列

3.5 MATLAB 的浮点数表达

计算数学模型的近似解时，影响较大的误差之一就是舍入误差，由于计算机使用有限精度的数来表示实际数值，因此产生了这种误差。

可以想像，要真正准确地写出实际数值需要消耗无限的纸和笔。例如，如下这个循环数：

$$\frac{8}{3} = 2.6666\cdots = \left(\frac{2}{10^1} + \frac{6}{10^2} + \frac{6}{10^3} + \frac{6}{10^4} + \frac{6}{10^5} + \cdots\right) \times 10^1$$

就是一个写不完的无限序列。但是，计算机只能用有限的内存空间表示实际数值，因此，无论采用何种表示方法，任何数据都只能包含有限位数。例如，把 8/3 这个无限序列截断为含有 4 位小数的有限数，就得到

$$\frac{8}{3}=2.6666\cdots=\left(\frac{2}{10^1}+\frac{6}{10^2}+\frac{6}{10^3}+\frac{6}{10^4}\right)\times 10^1=0.2666\times 10^1$$

每个实际数值 x 都有其浮点数表示，记为 $fl(x)$，假设浮点数有 t 位小数（称为精度），则

$$fl(x)=\pm 0.d_1d_2d_3\cdots d_{t-1}d_t\times 10^e$$
$$=\pm\left(\frac{d_1}{10^1}+\frac{d_2}{10^2}+\frac{d_3}{10^3}+\cdots+\frac{d_{t-1}}{10^{t-1}}+\frac{d_t}{10^t}\right)\times 10^e$$

确定式中正负号以及数字 d_i 和指数 e 的值，使 $fl(x)$ 尽可能接近 x 的实际数值。一个数的浮点数表示并不是惟一的，比如 0.2666×10^1 可以表示为 0.02666×10^2。

规格化的浮点数表示法规定 $d_1\neq 0$，因此，有 $1\leqslant d_1\leqslant 9$，并且对于 $i=2,\cdots,t$，有 $0\leqslant d_i\leqslant 9$。指数的范围也有限制，在给定的浮点数系统中，总是存在两个整数 $U>0$ 和 $L<0$，所有指数必须满足不等式 $L\leqslant e\leqslant U$。由此可见，系统能表示的最大数为

$$0.99\cdots 99\times 10^U<10^U$$

能表示的最小正数就是

$$0.10\cdots 00\times 10^{L-1}$$

3.5.1 浮点数表达的 IEEE 754 标准

如今绝大多数计算机使用二进制数，遵循 1985 年建立的 IEEE 标准（即 IEEE754）。其中，每位数字 d_i 的值只能取 0 或 1，这样，对于规格化的浮点数表示，第一位数字必定为 $d_1=1$，就没有必要存储了。IEEE 标准的二进制浮点数表示为

$$fl(x)=\pm\left(1+\frac{d_1}{2}+\frac{d_2}{4}+\frac{d_3}{8}+\cdots+\frac{d_t}{2^{t-1}}\right)\times 2^e$$

MATLAB 中使用的 IEEE 标准浮点数为 64 位字长，称为双精度数。64 位中，第一位是符号位 s，若 $s=0$，表示正数；若 $s=1$，则表示负数。随后的 11 位是指数，再后面的 52 位是小数。小数有 52 位，加上第一位 1，因此，其精度是 $52+1=53$ 位。

指数没有特定的符号位，指数的正负用一种偏移的方法来实现，也就是存储的指数值等于实际指数加一个偏移量。对于 IEEE 双精度浮点数，这个偏移量为 1023。因此，如果指数域存放的值为 1023，则表示实际指数值为 0；如果存放的值为 2000，则表示实际指数值为 977（即 2000 − 1023）。

对于双精度数，可以表示的最大数约为 10^{308}，最小正数约为 2.2×10^{-308}。

例 3.3 IEEE 754 浮点数表达

以下 64 位二进制数是一个 IEEE754 双精度浮点数：

0100000001111110100000000000000000000000000000000000000000000000

解释如下：

1）第一位符号位为 0，所以这是正数；

2）第二位开始的 11 位数字“10000000111”是指数 e，这 11 位二进制数对应的十进制数值为 1031，因此，实际指数的值为 e = 1031 - 1023 = 8。

3）再后面的 52 位数字“1110100000000000000000000000000000000000000000000000”是浮点数的小数部分，也称为尾数。这个尾数的值为 0.90625。

因此，这个浮点数的十进制数值为

$$符号\cdot(1+尾数)\times 2^{指数}$$

即 $1.90625\times 2^8 = 488$

指数值 -1023[○]（即所有位均为 0）和值 1024（即所有位均为 1）保留用于表示特殊的数值。由于第一位总是 1，真正的数值零无法直接表示。因此，设定零是一个特殊的值，用指数域和小数域的二进制位都为 0 来表示。这样，就有了 -0 和 +0 两个不同的零值，但是，作比较时，两者的值是相等的。

无穷大也是特殊数值，“+∞”和“-∞”这两个值用所有指数位为 1，所有小数位为 0 来表示，符号位用于区分无穷大的正负。把无穷大表示为特殊值，很有用，可以处理溢出情况。

值“NaN”（Not a Number）用于表示“不是数”，其所有指数位为 1，而小数不等于 0。NaN 可用于检测 0 除 0 等异常状态，出现这类错误时，用“NaN”来处理，可以不中止程序的运行。

如果一个数太大，超出了浮点数系统可以表示的范围，称为“上溢”。如果一个数的指数小于最小可以表示的值（在 IEEE 754 的双精度数系统中是 -1023），则称为“下溢”。计算过程中如果出现上溢，通常是致命的错误；而下溢没什么关系，系统通常把下溢的数置为 0，就继续运行程序（MATLAB 就这么处理，不给出警告信息）。

3.5.2　浮点数的算术运算、截断和舍入

用有限的 t 位数字存放实际数值有两种方法，一种是截断，忽略后面的位 d_{t+1}、d_{t+2}、d_{t+3} 等等；另一种是舍入，当 $d_{t+1}\geq 5$ 时，d_t 加 1。例如，假设 $t=3$，则有

○ 原书此处为 1023。——译者注

x	截　断	舍　入
5.672	5.67	5.67
-5.672	-5.67	-5.67
5.677	5.67	5.68
-5.677	-5.67	-5.68

数值 10^{1-t}称为系统的误差限（eps），是单次计算引入的最小误差量。对于IEEE 标准的双精度浮点数，系统的误差限为 2.2×10^{-16}。在 MATLAB 指令窗中键入 eps，所显示的值差不多就是 2.2×10^{-16}。值 $t-1$ 通常称为有效数字位数。

即使 IEEE 754 系统的数值表示是准确无误的，数的算术运算也会引入舍入误差。IEEE 标准要求运算过程中使用一种称为精确舍入的方法，也就是必须先精确计算，然后再舍入为最接近的浮点数。假设 $\mathrm{fl}(x)$ 和 $\mathrm{fl}(y)$ 为两个浮点数，则精确舍入就是

$$\mathrm{fl}(\mathrm{fl}(x)\pm\mathrm{fl}(y)) = (\mathrm{fl}(x)\pm\mathrm{fl}(y))(1+\varepsilon_1)$$
$$\mathrm{fl}(\mathrm{fl}(x)\times\mathrm{fl}(y)) = (\mathrm{fl}(x)\times\mathrm{fl}(y))(1+\varepsilon_2)$$
$$\mathrm{fl}(\mathrm{fl}(x)\div\mathrm{fl}(y)) = (\mathrm{fl}(x)\div\mathrm{fl}(y))(1+\varepsilon_3)$$

其中，$\varepsilon_i\leqslant eps$。用精确舍入法，可以保证每次运算后引入的相对误差最小。

例 3.4　浮点数误差的传播

假设 $x=0.1103$，并且 $y=0.9963\times10^{-2}$，则两数之差的准确值为 $x-y=0.100337$。如果截取精度都是 4 位（即 $t=4$），请分别计算以下两种计算方式下两数之差的相对误差：（1）用两数之差的准确值截取；（2）截取浮点数之后计算两数之差。

解：

（1）截取两数之差准确值的前 4 位，其值为 0.1003[○]，因此，精确舍入的相对误差为

$$\frac{|0.100337-0.1003|}{|0.100337|}\approx0.37\times10^{-3}$$

（2）如果将浮点数先截成 4 位，再求两数之差，则其结果的相对误差为

$$\frac{|0.100337-0.1004|}{|0.100337|}\approx0.63\times10^{-3}$$

显然，第二个相对误差已经超过了浮点数系统的舍入单位，即 $\frac{1}{2}\times10^{-3}$。

○ 原文此处为 0.1033。——译者注

例 3.5　MATLAB 系统的精度

通常，对于任意的 α 值，若 $\alpha \leqslant eps$，则 $\mathrm{fl}(1+\alpha)=1$。在 MATLAB 中，如下指令：

```
gamma = .5 * 2^( -100)
delta = (1 + gamma) - 1
```

产生的输出结果为

```
gamma = 3.9443e - 031
delta = 0
```

请利用这个输出结果，解释为什么例 3.1 中，对于很小的 h 值，误差曲线是平的？

解：

如图 3.2 所示，对于很小的 h 值，舍入误差为主。本例题的 MATLAB 程序显示，这种很小的 h 值（即程序中的 gamma），是加不上，也减不掉的。因此，在这个 h 值或更小的 h 值处估计的导数值总是 0，误差也就是常数。

3.5.3　舍入误差的累积和消去误差

数值算法的很多运算都不可避免地会引入小误差，要认识到每步浮点数运算都会带入小的相对误差，并且更重要的是要了解这些误差的累积情况。下面是几条规则：

1）如果 x 和 y 的大小差别很大，则 $x+y$ 的绝对误差会很大。

2）如果 $|y| \ll 1$，则 x/y 有很大的相对误差和绝对误差。如果 $|y| \gg 1$，则 xy 也会有很大的相对误差和绝对误差。

3）如果 $x \approx y$，则 $x-y$ 有很大的相对误差，这种误差称为消去误差。

4）上溢会导致计算结果出错，但有时可以避免。

例 3.6　避免上溢

假设浮点数系统的精度为 4 位，指数为 2 位。如果 $a=10^{60}$，且 $b=1$，计算 $c=\sqrt{a^2+b^2}$ 的值。这个精度下的正确答案是 $c=10^{60}$。请说明，在计算过程中，如何用事先调节比例的方法避免上溢。

解：

对任何 $s \neq 0$，都有 $c=s\sqrt{(a/s)^2+(b/s)^2}$ 成立，设 $s=a=10^{60}$，则计算 b/s 的平方时，会产生下溢，也就是得到的值为 0。这样，就可以求得 c 值的正确答案。

消去误差在实际计算过程中经常出现，了解其存在很重要。例如，在例 3.1 中，光滑（可求导）函数 $f(x)$ 在 $x = x_0$ 处的导数用两个相邻点的 f 函数值之差除以自变量的差来近似，即：

$$f'(x_0) \approx \frac{f(x_0 + h) - f(x_0)}{h}$$

当 h 很小时，分子存在消去误差，并且会被分母放大。算法上进行简单的修改，有时就可以避免这种误差。

例 3.7 避免消去误差

设 $x = 100000$，并且浮点数系统的精度为 5 位十进制数，请计算 $y = \sqrt{x+1} - \sqrt{x}$ 的值。该浮点数系统不能准确表示 100001 这个数，无论是截断还是舍入，在该系统中，这个数都是 100000。也就是说，在该浮点数系统下，对于 $x = 100000$，有 $x + 1 = x$。盲目地求 $\sqrt{x+1} - \sqrt{x}$ 的值，结果只能为 0。请设计一种能够计算出该表达式正确值的方法。

解：

以下恒等式可用于求 $\sqrt{x+1} - \sqrt{x}$ 的值：

$$\frac{(\sqrt{x+1} - \sqrt{x})(\sqrt{x+1} + \sqrt{x})}{(\sqrt{x+1} + \sqrt{x})} = \frac{1}{(\sqrt{x+1} + \sqrt{x})}$$

对于 $x = 100000$，结果为 0.15811×10^{-2}，就是 5 位十进制数的准确值。

还有一种避免消去误差的方法就是利用泰勒级数展开式。

例 3.8 利用泰勒级数展开式避免消去误差

对于很小的正数 x，即 $x > 0$，且 $|x| \approx 0$。请说明如何计算 $y = \sinh(x) = (e^x - e^{-x})/2$ 的值，才能使减法引起的消去误差最小。

解：

利用如下泰勒级数展开式

$$\sinh(x) = x + \frac{x^3}{6} + \frac{x^5}{120} + \cdots$$

可以在计算 y 值时，避免消去误差，因为泰勒级数中只有加法运算，没有减法运算。如果 x 很小，用截断级数 $x + \frac{x^3}{6}$ 就可以近似 y 值，因为 $|x| \leqslant 1$ 时，截断误差只有 $\approx \frac{x^5}{120}$。假设采用与例 3.7 相同的 5 位十进制数系统，则 $\sinh(0.1) = 0.10017$，而 $\sinh(0.01) = 0.01$。这些是该浮点数系统下的“准确”值。如果用没有离散化误差的指数函数进行计算，则 $x = 0.1$ 和 $x = 0.01$ 时的 y 值分别为 0.10018 和 0.010025。

设 $z = x - y$，其中 $x \approx y$，那么，用如下数学方法可以考察浮点数系统中这两

个几乎相等的数相减产生的效果：

$$|z - \mathrm{fl}(z)| \leqslant |x - \mathrm{fl}(x)| + |y - \mathrm{fl}(y)|$$

相对误差为

$$\frac{|z - \mathrm{fl}(z)|}{|z|} \leqslant \frac{|x - \mathrm{fl}(x)| + |y - \mathrm{fl}(y)|}{|x - y|}$$

因为 $x \approx y$，式中的分母接近于 0，所以，相对误差就可能很大。

使用舍入算法时，误差会有正负极性变化，而截断算法则没有。因此，在统计意义上，大规模的计算中，舍入算法的误差之间可能偶尔会互相抵消。

3.6　本章学习要点

学完本章之后，读者应该掌握以下内容：

1）科学计算的误差来源有：数学模型本身的误差、输入数据的测量误差、求连续函数近似解引起的误差和各种计算中的舍入误差等。

2）泰勒级数和泰勒公式是连续函数近似解数值算法的开发工具。

3）必须减小误差，使数值解接近准确解。

4）绝对误差由截断误差和舍入误差两部分组成。当截断误差较小时，舍入误差较大；反之，当截断误差较大时，舍入误差就较小。这是数值方法中最重要的折中。

5）浮点数是计算机表示的实数，它本身就存在误差，要么由截断引起，要么由舍入引起。

6）MATLAB 的浮点数表示法遵循 IEEE 754 标准。

7）转化计算公式的形式常常可以避免消去误差。

3.7　习题

3.1　利用如下三角恒等式：

$$\cos(\phi) - \cos(\varphi) = -2\sin\left(\frac{\phi+\varphi}{2}\right)\sin\left(\frac{\phi-\varphi}{2}\right)$$

可以把函数 $f_1(x,\delta) = \cos(x+\delta) - \cos(x)$ 转化为另一种 $f_2(x,\delta)$ 的形式。如果算术运算完全准确，那么，对于任意给定的 x 和 δ 值，f_1 和 f_2 的值相等。

请编写一个 MATLAB 脚本，当 $x = 3$，且 $\delta = 1.0\mathrm{e}^{-11}$ 时，计算 $g_1(x,\delta) = f_1(x,\delta)/\delta + \sin(x)$ 和 $g_2(x,\delta) = f_2(x,\delta)/\delta + \sin(x)$ 的值，并解释两个计算结果之间的差别。

3.2　利用如下三角公式：

$$\sin(\phi) - \sin(\varphi) = 2\cos\left(\frac{\phi+\varphi}{2}\right)\sin\left(\frac{\phi-\varphi}{2}\right)$$

可以把函数$f_1(x_0,h)=\sin(x_0+h)-\sin(x_0)$转化为另一种$f_2(x_0,h)$的形式。如果算术运算完全准确，那么，对于任意给定的x_0和h值，f_1和f_2的值相等。

请找出一个避免消去误差的公式，用于计算$f(x)=\sin(x)$在$x=x_0$处导数的近似值$\frac{f(x_0+h)-f(x_0)}{h}$。并编写 MATLAB 程序实现这个算法，计算$h=1\times10^{-20}$，…，1 时的$f'(0.5)$的近似值。并将计算结果与例 3.1 的结果进行比较，解释这两个结果之间的差别。

3.3 在只有t位有限精度的十进制浮点数系统中，不能完全准确地表示$\frac{8}{3}=2.666\cdots$。请问是否存在一个有限精度的浮点数系统确实可以准确表示这个数？如果是，请描述这样的系统。对于无理数π，这个问题的答案又是怎样？

3.4 当$b^2>c$时，二次方程式$x^2-2bx+c=0$的根为

$$x_{1,2}=b\pm\sqrt{b^2-c}$$

注意，$x_1x_2=c$。以下两个 MATLAB 脚本用两种不同的方法计算这两个根：

```
a) x1 = b + sqrt(b^2 - c);
   x2 = b - sqrt(b^2 - c);
b) if b > 0
   x1 = b + sqrt(b^2 - c);
   x2 = c / x1;
   else
   x2 = b - sqrt(b^2 - c);
   x1 = c / x2;
   end
```

通常哪个算法给出的结果更准确？不要计算，试回答该问题。

3.5 请编写 MATLAB 程序完成以下计算：

a）当$n=1$，2，…，100000 时，求$\frac{1}{n}$的和；

b）当$n=1$，2，…，100000 时，先将每个$\frac{1}{n}$的值舍入成为 4 位十进制数，然后再用 5 位十进制数的算术运算求这些数的和；

c）以相反的顺序，即$n=100000$，…，2，1，计算与 b）相同的舍入数之和。

比较并解释这 3 种计算的结果。

3.6 在统计学中（参见第 9 章），经常需要计算$\bar{x}=\frac{1}{n}\sum_{i=1}^{n}x_i$和$s^2=\frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x})^2$这两个量，其中$x_1$，$x_2$，…，$x_n$为给定的数据。容易看出$s^2$也可以写为$s^2=\frac{1}{n}\sum_{i=1}^{n}x_i^2-\bar{x}^2$。

a）s^2的这两种算法中哪一种的运算时间较少？假设在计数运算之前，$\bar{x}$的值已求得。

b）通常哪一种算法给出的s^2值更准确？

c）请编写 MATLAB 脚本，验证以上问题的答案是否正确。

3.7　利用本章 3.5 节所述的精确舍入，每个初等运算的相对误差都限制在舍入误差限之内，这个误差限在 MATLAB 中称为 eps。现在，考虑指数运算，算法为 $x^y = e^{y\ln x}$。请估计浮点数系统中 x^y 计算的相对误差。假设 $fl(x+y) = (x+y)(1+\varepsilon)$，其中 $|\varepsilon| \leq eps$，并假设其他操作都没有误差。并请说明，当 x 很大时，初等运算的误差限对于指数运算不成立。

3.8　参考文献

Atkinson, K. A. 1989. *An Introduction to Numerical Analysis*. New York: John Wiley & Sons.

Burden , R. L. and Faires, J. D. 1993. *Numerical Analysis*. Boston, MA: PWS Kent.

Chapra, S. C. and Canale, R. P. 2006. *Numerical Methods for Engineers*. 5th ed. Boston, MA: McGraw Hill Book Company.

Hoffman, J. D. 2001. *Numerical Methods for Engineers and Scientists*. New York: Marcel Dekker.

第2部分　系统的稳态行为

第4章　生物医学系统的线性模型

4.1　绪论

本章讲述求解线性系统方程的数值方法。如下线性代数方程组

$$
\begin{aligned}
a_{11}x_1 + a_{12}x_2 + \cdots\cdots &= b_1 \\
a_{21}x_1 + a_{22}x_2 + \cdots\cdots &= b_2 \\
\cdots\cdots\cdots\cdots\cdots\cdots\cdots & \\
a_{n1}x_1 + a_{n2}x_2 + \cdots\cdots &= b_n
\end{aligned}
\tag{4.1}
$$

可以用矩阵形式表示为 $\boldsymbol{Ax}=\boldsymbol{b}$，其中，$\boldsymbol{A}$ 为系数矩阵，$\boldsymbol{x}$ 是未知数矢量，$\boldsymbol{b}$ 是常数矢量（见式（4.2））。若系数矩阵 $\boldsymbol{A}$ 由常数 a_{ij}组成，不是 $\boldsymbol{x}$ 的函数，则方程组是线性的。若 $\boldsymbol{b}=0$，则方程组是齐次的。

$$
\boldsymbol{A}=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix};\quad \boldsymbol{x}=\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix};\quad \boldsymbol{b}=\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \tag{4.2}
$$

本章主要包括如下学习内容：

1）学习线性代数方程组数值求解的理论基础。

2）开发 MATLAB 程序，用两类不同的方法，即非迭代法（高斯法、高斯—若尔当法等）和迭代法（雅可比法、高斯—赛德尔法等），求解 $\boldsymbol{Ax}=\boldsymbol{b}$ 形式的问题。

3）用线性方程组 $\boldsymbol{Ax}=\boldsymbol{b}$ 描述生物医学系统，并用 MATLAB 程序求解这些方程组。

4.2　线性生物医学系统举例

许多生物医学领域的重要问题可以用线性系统来描述，本章将列举如下 3 方面具有代表性的线性系统：生物力学、生物医学图像、代谢工程和细胞生物技术。

4.2.1　生物力学中的力平衡

虽然生物力学领域涉及各种复杂的非线性系统，但是，在静力学中也普遍存在线性系统。力的线性分解是静力学的基础，力可以用标量分量和单位矢量表示。二维力矢量 $\boldsymbol{F}$ 由 x 方向的分量 F_x 和 y 方向的分量 F_y 组成，即

$$\boldsymbol{F} = F_x\boldsymbol{i} + F_y\boldsymbol{j} \tag{4.3}$$

式中　$\boldsymbol{i}$, $\boldsymbol{j}$——x 方向和 y 方向的单位矢量。

矢量的加就是求各个分量之和。矢量的乘积有两种：点积和叉积。点积的物理意义是一个矢量在另一个矢量上的投射；叉积是产生一个新矢量，该矢量的方向由右手螺旋法则确定。

对于生物医学系统的静力学平衡问题，利用牛顿运动方程可以建立如下力 $\boldsymbol{F}$ 力矩 $\boldsymbol{M}$ 的矢量方程：

$$\sum \boldsymbol{F} = 0 \tag{4.4}$$

$$\sum \boldsymbol{M} = 0 \tag{4.5}$$

生物力学中的一个典型问题是用自由体图描述人体需要研究部位所受到的外力作用以及反应力的情况。图 4.1 所示的是一个简单的自由体图，表示伸出的手臂的重量施加在肩膀上的力。

图中肌肉和关节产生的力可以分解成 x 和 y 方向的分量。由静力学平衡条件可知，x 方向和 y 方向的力分量可以求和，并且 z 方向的力矩守恒，由此可以建立一个线性代数方程组模拟该系统。用本章介绍的数值方法求解该方程组，就可以求得肌肉和关节产生的力中包含的未知量。图 4.1 中下方所示就是自由体图，其中，$\boldsymbol{F}_{关节}$ 分解为 $\boldsymbol{F}_3$ 和 $\boldsymbol{F}_4$ 两个分量；$\boldsymbol{F}_{肌肉}$ 用 $\boldsymbol{F}_2$ 表示；$\boldsymbol{F}_{重力}$ 用 $\boldsymbol{F}_1$ 表示。

由 x 方向和 y 方向的静力平衡以及 z 方向的力矩平衡可得

$$\boldsymbol{F}_2\cos\alpha + \boldsymbol{F}_3 = 0 \qquad (x\ 方向) \tag{4.6}$$

$$-\boldsymbol{F}_1 + \boldsymbol{F}_2\sin\alpha + \boldsymbol{F}_4 = 0 \qquad (y\ 方向) \tag{4.7}$$

$$r_1\boldsymbol{F}_1 - r_2\boldsymbol{F}_2\sin\alpha = 0 \qquad (z\ 方向) \tag{4.8}$$

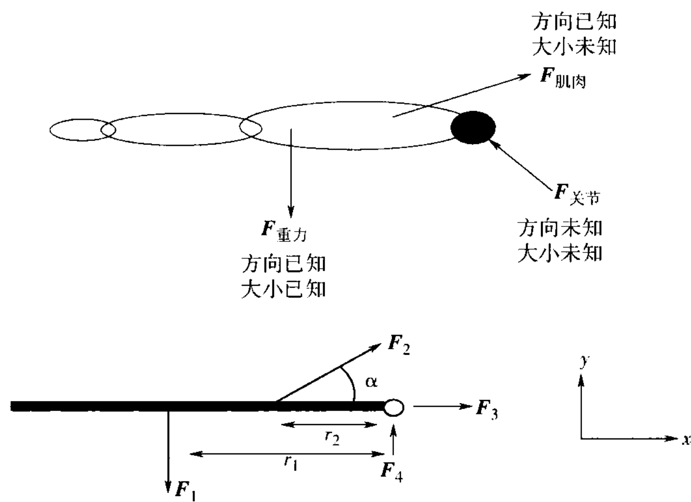


图 4.1　人体上力作用的自由体图

如果写成矩阵形式，该线性代数方程组就是一个 3 ×3 矩阵系统，可以求解。对于较复杂的生物力学系统，模型中就会有更多的方程需要联立求解。本章就是要讲述求解这些方程组的数值方法。

4.2.2　生物医学图像以及图像处理

生物医学图像处理也要用到线性方程组，尤其是计算机 X 射线断层扫描成像（CT）中的三维图像重建，可以用代数方法实现图像重建。

例如，在未知图像$f(x, y)$上加方形网格（见图 4.2）。假设网格中每个单元j的图像函数值f_j为常数。射线束i扫描整个(x, y)平面，射线束的宽度τ与网格单元宽度基本相等。p_i表示第i射线束的投影值，则各个网格单元f_j对p_i的贡献可以表示为

$$\sum_{j=1}^{N} w_{ij} f_j = p_i, \qquad i = 1,2,\cdots,M \tag{4.9}$$

式中　w_{ij}——权重因子，表示第j个网格单元对于第i射线束投射值的贡献；

N——网格单元数目；

M——射线束的扫描投影数。

M和N较小时，系统模型较简单，用数值方法求矩阵的逆可以解这个方程组，本章后面将讲述这些数值方法。但是，实际图像即使只有 256 ×256 像素，N值也大于 50 000，因此，必须采用简化方法重建模型。

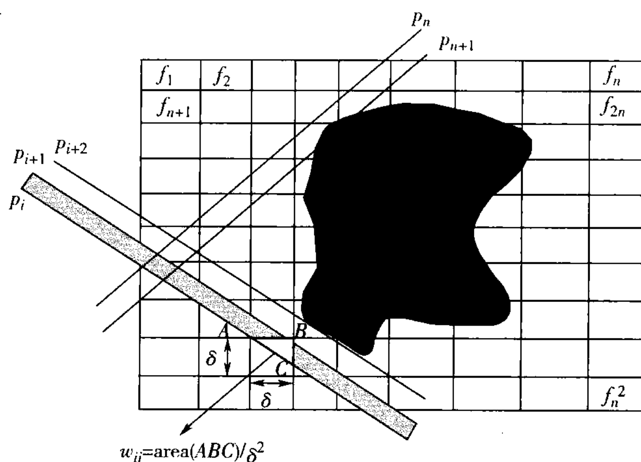
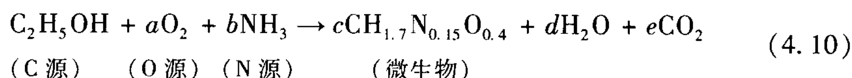


图 4.2　生物医学图像的重建（摘自 Rosenfeld 和 Kak，1982）

4.2.3　代谢工程和细胞生物技术

在生物技术和代谢工程中需要对微生物细胞的培养和生长进行设计（Shuler 和 Kargi，2002），也就是要设计生物反应器。生物反应器利用生化底物为微生物的生长提供基本化学元素。以下是一个表示微生物需氧生长过程的典型化学反应式：



$$\underset{(\text{C 源})}{C_2H_5OH} + \underset{(\text{O 源})}{aO_2} + \underset{(\text{N 源})}{bNH_3} \rightarrow \underset{(\text{微生物})}{cCH_{1.7}N_{0.15}O_{0.4}} + dH_2O + eCO_2 \tag{4.10}$$

每消耗单位摩尔氧气产生的二氧化氮摩尔数称为呼吸熵（RQ），也就是式中系数 e 与 a 的比率，即 $e = (RQ)\,a$。RQ 的值可以由实验测定，因此可以作为已知量。为了从理论上计算微生物生长繁殖的产量，还必须知道这个化学反应式中 a、b、c 和 d 各系数的值。这可以通过建立反应式包含的四大元素的物质平衡方程来求得。

各元素的物质平衡方程为

$$\begin{aligned}
&\text{C(碳)}: 2 &&= c + (RQ)a \\
&\text{H(氢)}: 6 + 3b &&= 1.7c + 2d \\
&\text{O(氧)}: 1 + 2a &&= 0.4c + d + 2(RQ)a \\
&\text{N(氮)}: b &&= 0.15c
\end{aligned} \tag{4.11}$$

写成矩阵形式：

$$\begin{bmatrix} RQ & 0 & 1 & 0 \\ 0 & -3 & 1.7 & 2 \\ -2 & 0 & 0.4 & 1 \\ 0 & -1 & 0.15 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} 2 \\ 6 \\ 1-2RQ \\ 0 \end{bmatrix} \tag{4.12}$$

式（4.11）及其相应的矩阵形式的式（4.12）是包含4个未知数的4阶线性代数方程组。本章将讲述这类问题求解的数值方法。

4.3 线性代数方程组

求解线性代数方程组最常用的方法是高斯消元法，其基本原理是将给定的 n 阶“全”线性代数方程组变换为三角形式的 n 阶方程组，方程组的解也就随即可得。举例如下。

4.3.1 3×3 阶矩阵的简单高斯消元法示例

如下为3×3阶系统方程组的一个例子：

$$\begin{aligned} 2x + y - z &= 7 \\ 2x + 6y + 5z &= 0 \\ 3x + y + z &= 5 \end{aligned} \tag{4.13}$$

高斯消元法可分为两个简单步骤：

步骤1：利用第一个方程，消去第二、三个方程中的 x 项：

- 将第一个方程乘以（−1），加到第二个方程上，生成不含 x 变量的第二个新方程。
- 将第一个方程乘以（−3/2），并加到第三个方程上，得到不含 x 变量的第三个新方程。

于是，方程组转化成为

$$\begin{aligned} 2x + y - z &= 7 \\ 5y + 6z &= -7 \\ -\frac{1}{2}y + \frac{5}{2}z &= -\frac{11}{2} \end{aligned} \tag{4.14}$$

步骤2：利用第二个方程，消去第三个方程中的 y 项：

- 将第二个方程乘以（1/10），加到第三个方程上，把第三个方程变换为不含 y 变量的新方程。

于是，方程组变为

$$\begin{aligned} 2x + y - z &= 7 \\ 5y + 6z &= -7 \\ \frac{31}{10}z &= -\frac{31}{5} \end{aligned} \tag{4.15}$$

这样，就完成了“向前消元法”，得到的是一个上三角矩阵。变换后的矩阵 $\boldsymbol{A}$ 记为 $\boldsymbol{U}$。于是，原始方程 $\boldsymbol{Ax}=\boldsymbol{b}$ 被变换为

$$\begin{bmatrix} 2 & 1 & -1 \\ 0 & 5 & 6 \\ 0 & 0 & \frac{31}{10} \end{bmatrix}\begin{bmatrix} x \\ y \\ z \end{bmatrix}=\begin{bmatrix} 7 \\ -7 \\ -\frac{31}{5} \end{bmatrix}$$

即

$$\boldsymbol{Ux}=\bar{\boldsymbol{b}} \tag{4.16}$$

由方程组（4.15）或（4.16）很容易得到解 $z=-2$，经过代换，即可得 $y=1$，$x=2$。

4.3.2　高斯消元法的矩阵表示

线性代数方程组的通用矩阵形式为 $\boldsymbol{Ax}=\boldsymbol{b}$。下面将上述高斯消元法推广应用到这种包含系数矢量 $\boldsymbol{b}$ 的一般线性代数方程组。将变量去除，留下只包含系数的矩阵，这种矩阵称为增广矩阵。上例式（4.13）方程的原始增广矩阵为

$$\left[\begin{array}{ccc|c} 2 & 1 & -1 & 7 \\ 2 & 6 & 5 & 0 \\ 3 & 1 & 1 & 5 \end{array}\right]$$

而变换之后，其最终的增广矩阵为

$$\left[\begin{array}{ccc|c} 2 & 1 & -1 & 7 \\ 0 & 5 & 6 & -7 \\ 0 & 0 & \frac{31}{10} & -\frac{31}{5} \end{array}\right] \tag{4.17}$$

现在来归纳高斯消元法矩阵运算的一般算法步骤。

1. 高斯增广矩阵的定义

在矩阵变换开始之前，先把矩阵 $\boldsymbol{A}$ 和常数矢量 $\boldsymbol{b}$ 组合成增广矩阵，假设这一步为第 0 步（$k=0$），即

$$\begin{aligned} &\text{For}\quad i=1,2,\cdots,n && \text{(所有的行)} \\ &a_{ij}^{(k=0)}=a_{ij}\quad j=1,2,\cdots,n && \\ &a_{ij}^{(k=0)}=c_{ij}\quad j=n+1 && \text{(扩增的列)} \end{aligned} \tag{4.18}$$

2. 高斯消元法

下面逐步分析矩阵变换过程。对角线上的元素称为主元素。

第一步变换（即 $k=1$）：将原始 R_2 行减去 R_1 行（$i=1$）乘以一个乘数，得到新的变换行 $R_2^{k=1}$。可以看出这个乘数是原始 R_2 行第一个非 0 系数与 R_1

行主元素之比。R_2 行的每列元素计算完成之后，再按照此方法计算 R_3 行，将原始 R_3 行减去 R_1 行乘以一个乘数，得到新的变换行 $R_3^{k=1}$，但此时这个乘数是原始 R_3 行第一个非 0 系数与 R_1 行主元素之比。依此类推，计算以后的各行。可见，在第 k 步变换中，需要进行变换的行是第 $(k+1)$ 行及其之后的所有行。

对于这个例子，第一步 $(k=1)$ 第二行的变换结果为

$$\begin{array}{c} \\ \\ R_1^{k=0}\rightarrow \\ R_2^{k=1}\rightarrow \\ R_3^{k=0}\rightarrow \end{array}\begin{array}{c} \begin{array}{cccc} j=1 & j=2 & j=3 & j=4 \\ \Downarrow & \Downarrow & \Downarrow & \Downarrow \end{array} \\ \left[\begin{array}{cccc} 2 & 1 & -1 & 7 \\ 2-\dfrac{2}{2}\times 2 & 6-\dfrac{2}{2}\times 1 & 5-\dfrac{2}{2}\times(-1) & 0-\dfrac{2}{2}\times 7 \\ 3 & 1 & 1 & 5 \end{array}\right] \end{array} \tag{4.19}$$

可见，第二行 $(i=2)$ 元素的变换结果可以用如下通式表示：

$$a_{2j}^{k=1}=a_{2j}^{k=0}-\frac{a_{21}^{k=0}}{a_{11}^{k=0}}a_{1j}^{k=0};\quad i=2;j=4,3,\cdots,k;k=1 \tag{4.20}$$

其中，乘数为 2/2。

第三行 $(i=3)$ 元素的变换结果为

$$\begin{array}{c} \\ \\ R_1^{k=0}\rightarrow \\ R_2^{k=1}\rightarrow \\ R_3^{k=1}\rightarrow \end{array}\begin{array}{c} \begin{array}{cccc} j=1 & j=2 & j=3 & j=4 \\ \Downarrow & \Downarrow & \Downarrow & \Downarrow \end{array} \\ \left[\begin{array}{cccc} 2 & 1 & -1 & 7 \\ 0 & 5 & 6 & -7 \\ 3-\dfrac{3}{2}\times 2 & 1-\dfrac{3}{2}\times 1 & 1-\dfrac{3}{2}\times(-1) & 5-\dfrac{3}{2}\times 7 \end{array}\right] \end{array}$$

其通式表示：

$$a_{3j}^{k=1}=a_{3j}^{k=0}-\frac{a_{31}^{k=0}}{a_{11}^{k=0}}a_{1j}^{k=0};\quad i=3;j=4,3,\cdots,k;k=1 \tag{4.21}$$

其中，乘数为 3/2。

于是，第一步变换完成之后，增广矩阵变为

$$\begin{array}{c} R_1^{k=0}\rightarrow \\ R_2^{k=1}\rightarrow \\ R_3^{k=1}\rightarrow \end{array}\left[\begin{array}{ccc|c} 2 & 1 & -1 & 7 \\ 0 & 5 & 6 & -7 \\ 0 & -\dfrac{1}{2} & \dfrac{5}{2} & -\dfrac{11}{2} \end{array}\right] \tag{4.22}$$

然后，进行第二步 $(k=2)$ 变换，其结果为

$$
\begin{array}{l}
R_1^{k=0} \rightarrow \\
R_2^{k=1} \rightarrow \\
R_3^{k=2} \rightarrow
\end{array}
\left[\begin{array}{cccc}
2 & 1 & -1 & | \quad 7 \\
0 & 5 & 6 & | \quad -7 \\
0 & -\frac{1}{2}-\left(-\frac{1}{10}\right)\times 5 & \frac{5}{2}-\left(-\frac{1}{10}\right)\times 6 & | \quad -\frac{11}{2}-\left(-\frac{1}{10}\right)\times(-7)
\end{array}\right]
$$

第三行（$i=3$）元素变换结果的通式为

$$
a_{3j}^{k=2}=a_{3j}^{k=1}-\frac{a_{32}^{k=1}}{a_{22}^{k=1}}a_{2j}^{k=1};\quad i=3;j=4,3,\cdots,k;k=2 \tag{4.23}
$$

第二步变换完成之后，增广矩阵变为

$$
\begin{array}{l}
R_1^{k=0} \rightarrow \\
R_2^{k=1} \rightarrow \\
R_3^{k=2} \rightarrow
\end{array}
\left[\begin{array}{cccc}
2 & 1 & -1 & | \quad 7 \\
0 & 5 & 6 & | \quad 7 \\
0 & 0 & \frac{31}{10} & | \quad -\frac{62}{10}
\end{array}\right] \tag{4.24}
$$

将以上变换步骤结合起来，可以写成如下算法：

$$
\begin{array}{ll}
k=1,2,\cdots,n-1 & (\text{步骤循环}) \\
i=k+1,k+2,\cdots,n & (\text{行循环}) \\
j=n+1,n,\cdots,k & (\text{给定行的列循环}) \\
a_{ij}^{(k)}=a_{ij}^{(k-1)}-\frac{a_{ik}^{(k-1)}}{a_{kk}^{(k-1)}}a_{kj}^{(k-1)} & (\text{高斯变换});\quad a_{kk}^{(k-1)}\neq 0
\end{array} \tag{4.25}
$$

增广矩阵的变换完成之后，立即可以从最后一行（第 n 行）开始进行简单的回代计算，一直回代到第一行为止，得到方程的解。即

$$
x_n=\frac{a_{n,n+1}}{a_{n,n}}(\text{第 } n \text{ 行的回代计算})
$$

由于 $a_{ii}x_i+\sum_{j=i+1}^{n}a_{ij}x_j=a_{i,n+1}$，可以得到第 n 行之前各行的回代计算通用公式：

$$
x_i=\frac{a_{i,n+1}-\sum_{j=i+1}^{n}a_{ij}x_j}{a_{ii}};i=n-1,n-2,\cdots,1\ (\text{行循环,向后回代}) \tag{4.26}
$$

3. 有关高斯消元法

虽然高斯消元法是求解线性代数方程组的有效方法，但是这种方法有两个缺点。

第一个缺点是由方程组系数矩阵的性质引起的。如果遇到某行的主元素为 0，并且后面各行在这一列上的元素也都为 0，则方程组要么不相容，要么有重复解，不能得到惟一解。但是，如果遇到某行的主元素为 0，而后面各行在这一列上有非 0 元素，那么，向前高斯消元法需要改进之后才能用。一般将主元素为

0 的行与该列上系数值最大的行进行交换，这种操作称为选主元，这种改进方法称为部分选主元高斯消元法，下节讲述这种方法。部分选主元减小了除 0 的可能性，并利用最大主元，提高了高斯消元法的计算准确度。

还有一种方法是将整个矩阵重新排列，选出可能的最大主元，除了进行搜寻之外，也搜寻列，交换列及其相关的变量。这种包括行交换和列交换的选主方法称为完全选主元法。

高斯消元法的第二个缺点在方程组系数矩阵为病态矩阵时出现。典型的病态矩阵是，与矩阵其他主元的值相比，有些主元的值非常小。在高斯消元法执行过程中，除以一个很小的、接近于 0 的数，会导致很大的舍入误差以及误差传播。下面是一个例子：

$$\begin{aligned} 0.0001x_1 + x_2 &= 2 \\ x_1 + 2x_2 &= 5 \end{aligned} \tag{4.27}$$

如果用基本高斯消元法，就是将第一个方程除以主元 0.0001，然后用第二个方程减去这个除之后的结果，得到如下方程：

$$\begin{aligned} 0.0001x_1 + x_2 &= 2 \\ [2 - 1/0.0001]x_2 &= 5 - 2/0.0001 \end{aligned}$$

舍入处理以后，x_2 的近似值为 $x_2=2$，于是，由第一个方程可知 $x_1=0$。显然，这个解不理想。

如果交换方程行的次序，也就是用部分选主元法，如下所示，就可以用较大的主元值

$$\begin{aligned} x_1 + 2x_2 &= 5 \\ 0.0001x_1 + x_2 &= 2 \end{aligned} \tag{4.28}$$

现在，按照高斯消元法的步骤，把第一个方程乘以 0.0001，然后从第二个方程中减去，得到

$$\begin{aligned} x_1 + 2x_2 &= 5 \\ [1 - 2 \times 0.0001]x_2 &= 2 - 5 \times 0.0001 \end{aligned}$$

得到 $x_2=2$，回代到第一个方程，可得 $x_1=1$，解的结果就好多了。

下面是只进行行交换的部分选主元法的通用算法步骤：

1. 初始化

$a_{pj} = temp \quad p = k|a_{kk}|$

For $i = 1,2,\cdots,n$ （所有行）

　$a_{ij}^{(k=0)} = a_{ij} \quad j = 1,2,\cdots,n$

　$a_{ij}^{(k=0)} = c_{ij} \quad j = n+1$ （扩增的列）

End

2. 部分选主元以及向前高斯消元法的算法

```
For k = 1 to n - 1
    Pivot = |a_kk|                      (Pivot = 主元)
    p = k                               (p = 主元行)
    For i = k + 1 to n
        If |a_ik| > Pivot
            Pivot = |a_ik|              (更新主元)
            p = i                       (更新主元行)
        End
    End
    If(p > k)                           (k 行和 p 行交换)
        For j = 1,n + 1
            temp = a_kj
            a_kj = a_pj
            a_pj = temp
        End
    End
    For i = k + 1,k + 2,……,n            (行循环)
        For j = n + 1,n,……,k            (给定行的列循环)
```

$$a_{ij}^{(k)} = a_{ij}^{(k-1)} - \frac{a_{ik}^{(k-1)}}{a_{kk}^{(k-1)}} a_{kj}^{(k-1)} \quad \text{（高斯变换）} \tag{4.29}$$

```
        End
    End
End  (End of loop on k,the counter for iteration step)
```

3. 回代计算

$$x_n = \frac{a_{n,n+1}}{a_{n,n}} \quad \text{（第 } n \text{ 行回代计算）}$$

For $i = n-1, n-2, \cdots, 1$ （行循环，向后回代）

$$x_i = \frac{a_{i,n+1} - \sum_{j=i+1}^{n} a_{ij} x_j}{a_{ii}} \tag{4.30}$$

End

通常，利用对角占优矩阵（即 $|a_{ii}| > \sum_{j \neq i} |a_{ij}|$）可以较好地运用高斯消元法。

例 4.1　应用高斯消元法求解如下 $\boldsymbol{Ax} = \boldsymbol{b}$ 形式的线性代数方程组

$$\begin{bmatrix} 2 & 1 & -1 \\ 2 & 6 & 5 \\ 3 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 7 \\ 0 \\ 5 \end{bmatrix}$$

解:

MATLAB 程序如下:

```
% example4_1.m - This program poses Ax = b in matrix form and
% solves for x using the Gauss elimination method. It calls the
% function GAUSS.M to transform the augmented matrix and do
% back-substitution.

clc; clear all;
% Input data
% Matrix of coefficients
A =...
   [2, 1, -1
   2, 6, 5
   3, 1, 1];
% Vector of constants
b = [7; 0 ; 5];
% Solving the set of equations by Gauss elimination method
x = Gauss(A,b);
% Show the results
disp(' Results:')
fprintf(' x1 =% 4.2f\n x2 =% 4.2f\n x3 =% 4.2f\n',x)
```

下面是完成高斯消元法的 MATLAB 函数，由 Constantinides 和 Mostoufi (1999)开发。

```
function x = Gauss (A , c)
% GAUSS Solves a set of linear algebraic equations by the Gauss
%   elimination method.
%
%   GAUSS(A,C) finds unknowns of a set of linear algebraic
%   equations. A is the matrix of coefficients and C is the
%   vector of constants.
```

```
%
%   See also JORDAN, SEIDEL

%   (c) N. Mostoufi & A. Constantinides
% January 1, 1999

c = (c(:).')';   % Make sure it's a column vector

n = length(c);
[nr nc] = size(A);

% Check coefficient matrix and vector of constants
if nr ~= nc
  error('Coefficient matrix is not square.')
end
if nr ~= n
  error('Coefficient matrix and vector of constants do not have the
same length.')
end

% Check if the coefficient matrix is singular
if det(A) == 0
  fprintf('\n Rank = % 7.3g\n',rank(A))
  error('The coefficient matrix is singular.')
end
unit = diag(ones(1 , n)); % Unit matrix
order = [1:n];               % Order of unknowns
aug = [A c];                 % Augmented matrix

% Gauss elimination
for k = 1:n - 1
    pivot = abs(aug(k , k));
    prow = k;
    pcol = k;
```

```
% Locating the maximum pivot element
for row = k:n
    for col = k:n
      if abs(aug(row, col)) > pivot
        pivot = abs(aug(row, col));
        prow = row;
        pcol = col;
    end
  end
end

% Interchanging the rows
pr = unit;
tmp = pr(k, :);
pr(k,:) = pr(prow,:);
pr(prow,:) = tmp;
aug = pr * aug;

% Interchanging the columns
pc = unit;
tmp = pc(k,:);
pc(k,:) = pc(pcol,:);
pc(pcol,:) = tmp;
aug(1:n, 1:n) = aug(1:n, 1:n) * pc;
order = order * pc;       % Keep track of the column interchanges

% Reducing the elements below diagonal to zero in the column k
lk = unit;
for m = k + 1:n
    lk(m, k) = - aug(m, k) / aug(k, k);
  end
  aug = lk * aug;
end
x = zeros(n, 1);
% Back substitution
```

```
t(n)=aug(n, n+1)/aug(n, n);
x(order(n))=t(n);
for k=n -1:-1:1
    t(k)=(aug(k,n+1)-sum(aug(k,k+1:n).*t(k+1:n)))/aug(k,k);
    x(order(k))=t(k);
end
```

程序运行之后的输出结果为

```
    x1 =2.00
    x2 =1.00
    x3 = -2.00
```

4.4　高斯-若尔当消元法

高斯-若尔当（Gauss-Jordan）消元法是高斯消元法的一种修正。其矩阵对角线上的元素被归一化为 1，并且对角线上方和下方的元素均为 0，也就是把系数矩阵变换成单位矩阵。方程组标准的初始矩阵形式为

$$\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b} \tag{4.31}$$

高斯-若尔当消元法的结果为

$$\boldsymbol{I}\boldsymbol{x}=\tilde{\boldsymbol{b}} \tag{4.32}$$

显然，矢量 $\tilde{\boldsymbol{b}}$ 就是解矢量。

高斯-若尔当消元法的证明

这里，先利用前面已经用过的那个增广矩阵来考察系数矩阵是如何消减为单位矩阵的。为了简化问题，此处先不考虑选主元法的行交换和列交换，结合选主元法的高斯-若尔当消元法将在下节讲述。初始矩阵为

$$\left[\begin{array}{ccc|c} 2 & 1 & -1 & 7 \\ 2 & 6 & 5 & 0 \\ 3 & 1 & 1 & 5 \end{array}\right]$$

将第一行除以 2，使该行归一化：

$$\left[\begin{array}{ccc|c} 1 & 1/2 & -1/2 & 7/2 \\ 2 & 6 & 5 & 0 \\ 3 & 1 & 1 & 5 \end{array}\right]$$

将归一化的第一行乘以 2，并从第二行中减去：

$$\begin{bmatrix} 1 & 1/2 & -1/2 & | & 7/2 \\ 0 & 5 & 6 & | & -7 \\ 3 & 1 & 1 & | & 5 \end{bmatrix}$$

将归一化的第一行乘以 3，并从第三行中减去：

$$\begin{bmatrix} 1 & 1/2 & -1/2 & | & 7/2 \\ 0 & 5 & 6 & | & -7 \\ 0 & -1/2 & 5/2 & | & -11/2 \end{bmatrix}$$

将第二行除以 5，使该行归一化：

$$\begin{bmatrix} 1 & 1/2 & -1/2 & | & 7/2 \\ 0 & 1 & 6/5 & | & -7/5 \\ 0 & -1/2 & 5/2 & | & -11/2 \end{bmatrix}$$

将归一化的第二行乘以 1/2，并从第一行中减去：

$$\begin{bmatrix} 1 & 0 & -11/10 & | & 42/10 \\ 0 & 1 & 6/5 & | & -7/5 \\ 0 & -1/2 & 5/2 & | & -11/2 \end{bmatrix}$$

将归一化的第二行乘以 -1/2，并从第三行中减去：

$$\begin{bmatrix} 1 & 0 & -11/10 & | & 42/10 \\ 0 & 1 & 6/5 & | & -7/5 \\ 0 & 0 & 31/10 & | & -62/10 \end{bmatrix}$$

将第三行除以 31/10，使该行归一化为

$$\begin{bmatrix} 1 & 0 & -11/10 & | & 42/10 \\ 0 & 1 & 6/5 & | & -7/5 \\ 0 & 0 & 1 & | & -2 \end{bmatrix}$$

将归一化的第三行乘以 -11/10，并从第一行中减去：

$$\begin{bmatrix} 1 & 0 & 0 & | & 2 \\ 0 & 1 & 6/5 & | & -7/5 \\ 0 & 0 & 1 & | & -2 \end{bmatrix}$$

将归一化的第三行乘以 6/5，并从第二行中减去：

$$\begin{bmatrix} 1 & 0 & 0 & | & 2 \\ 0 & 1 & 0 & | & 1 \\ 0 & 0 & 1 & | & -2 \end{bmatrix}$$

于是，就得到了本题的解 $x_1=2$，$x_2=1$，$x_3=-2$。

例 4.2 利用高斯-若尔当消元法求解如下 $\boldsymbol{Ax}=\boldsymbol{b}$ 问题：

$$\begin{bmatrix} 2 & 1 & -1 \\ 2 & 6 & 5 \\ 3 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 7 \\ 0 \\ 5 \end{bmatrix}$$

解：

MATLAB 程序为

```
% example4_2.m - This program solves the system Ax = b using the
% function JORDAN.M.

clc; clear all;

% Matrix of coefficients
A = [2, 1, -1; 2, 6, 5; 3, 1, 1];
% Vector of constants
b = [7; 0 ; 5];

% Solution
X = Jordan(A,b)
```

下面是完成高斯-若尔当消元法的 MATLAB 函数：

```
function x = Jordan (A , c)
% JORDAN Solves a set of linear algebraic equations by the
%   Gauss-Jordan method.
%
%   JORDAN(A,C) finds unknowns of a set of linear algebraic
%   equations. A is the matrix of coefficients and C is the
%   vector of constants.
%
%   See also GAUSS, GSEIDEL

% (c) N. Mostoufi & A. Constantinides
% January 1, 1999

c = (c(:).')';     % Make sure it's a column vector
```

```
n = length(c);
[nr nc] = size(A);

% Check coefficient matrix and vector of constants
if nr ~= nc
    error('Coefficient matrix is not square.')
end
if nr ~= n
    error('Coefficient matrix and vector of constants do not have
the same length.')
end

% Check if the coefficient matrix is singular
if det(A) == 0
    fprintf('\n Rank = % 7.3g\n',rank(A))
    error('The coefficient matrix is singular.')
end

unit = diag(ones(1, n));    % Unit matrix
order = [1:n];              % Order of unknowns
aug = [A c];                % Augmented matrix

% Gauss-Jordan algorithm
for k = 1:n
    pivot = abs(aug(k, k));
    prow = k;
    pcol = k;

    % Locating the maximum pivot element
    for row = k:n
        for col = k:n
            if abs(aug(row, col)) > pivot
                pivot = abs(aug(row, col));
                prow = row;
                pcol = col;
```

```
            end
        end
    end

    % Interchanging the rows
    pr = unit;
    tmp = pr(k , :);
    pr(k ,:) = pr(prow ,:);
    pr(prow ,:) = tmp;
    aug = pr * aug;

    % Interchanging the columns
    pc = unit;
    tmp = pc(k ,:);
    pc(k ,:) = pc(pcol ,:);
    pc(pcol ,:) = tmp;
    aug(1:n , 1:n) = aug(1:n , 1:n) * pc;
    order = order * pc;  % Keep track of the column interchanges

    % Reducing the elements above and below diagonal to zero
    lk = unit;
    for m = 1:n
        if m == k
          lk(m , k) = 1 / aug(k , k);
        else
          lk(m , k) = - aug(m , k) / aug(k , k);
        end
    end
    aug = lk * aug;
end
x = zeros(n , 1);
%  Solution
for k = 1:n
    x(order(k)) = aug(k , n + 1);
end
```

该程序的运行结果为

```
x =
    2.0000
    1.0000
   -2.0000
```

4.5 线性系统求解的迭代法

如果方程组的矩阵是对角占优矩阵，也就是，矩阵对角线上每一个系数的绝对值大于同一行中其他系数绝对值之和，那么，可以用迭代法求解该线性方程组。重新考察本章前面所用的方程组

$$\begin{aligned} 2x + y - z &= 7 \\ 2x + 6y + 5z &= 0 \\ 3x + y + z &= 5 \end{aligned} \tag{4.33}$$

比较矩阵对角线上系数的绝对值与其他系数绝对值之和的大小：

$$\begin{aligned} |2| &= |1| + |-1| \\ |6| &< |2| + |5| \\ |1| &< |3| + |1| \end{aligned}$$

可见，这个方程组不是对角占优的，不能用下面将要讲述的迭代法来求解。

4.5.1 雅可比法

用反复进行的代换计算可以求解方程组 $\boldsymbol{Ax} = \boldsymbol{b}$，这种方法称为迭代法。其原理是：把 $\boldsymbol{Ax} = \boldsymbol{b}$ 转化为等价方程组 $\boldsymbol{x} = \boldsymbol{Px} + \boldsymbol{q}$，并生成一个近似值序列：$x^{(1)}$，$x^{(2)}$，…。每一次迭代都必须满足关系 $x^{(k)} = \boldsymbol{P}x^{(k-1)} + \boldsymbol{q}$。其基本思路是用方程组的第 i 个方程求解第 i 个变量。例如，对于 3×3 阶简单系统：

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 &= b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 &= b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 &= b_3 \end{aligned} \tag{4.34}$$

可以转化为一个 $\boldsymbol{x} = \boldsymbol{Px} + \boldsymbol{q}$ 方程组，其中 $\boldsymbol{P}$ 矩阵对角线上的元素均为 0，即

$$\begin{aligned} x_1^{(k)} &= -\frac{a_{12}}{a_{11}}x_2^{(k-1)} - \frac{a_{13}}{a_{11}}x_3^{(k-1)} + \frac{b_1}{a_{11}} \\ x_2^{(k)} &= -\frac{a_{21}}{a_{22}}x_1^{(k-1)} - \frac{a_{23}}{a_{22}}x_3^{(k-1)} + \frac{b_2}{a_{22}} \end{aligned}$$

$$x_3^{(k)} = -\frac{a_{31}}{a_{33}}x_1^{(k-1)} - \frac{a_{32}}{a_{33}}x_2^{(k-1)} + \frac{b_3}{a_{33}}$$

开始时，给定以上方程组右边 x_1、x_2、x_3 的一组估计值，用它们计算出左边新的 $\boldsymbol{x}$ 矢量；再用新的 x_1、x_2、x_3 值替代方程右边的未知量，算出下一组新值。如此反复计算，直到某次迭代得到的新 $\boldsymbol{x}$ 矢量的值与上一次的迭代结果之差的范数足够小，或者残差矢量的范数 $\|\boldsymbol{Ax}-\boldsymbol{b}\|$ 小于一个特定的允许误差，迭代终止。这种方法称为雅可比迭代法。

雅可比法的算法为

$$x_i^{(k+1)} = \frac{1}{A_{i,i}}\left(b_i - \sum_{\substack{j=1 \\ j\neq i}}^{N} A_{i,j}x_j^{(k)}\right) \tag{4.35}$$

由于雅可比法依赖于前面迭代步骤中所得到的估计值，应用矩阵代数可以将雅可比法归纳成为一种简化的算法。重新考虑方程组

$$\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{b} \tag{4.36}$$

根据矩阵 $\boldsymbol{A}$ 写出一个对角矩阵 $\boldsymbol{D}$，使其对角线元素等于 $\boldsymbol{A}$ 的对角线元素，那么，$(\boldsymbol{A}-\boldsymbol{D})$ 的对角线元素就等于 0。用如下等式代替 $\boldsymbol{A}$：

$$\boldsymbol{A} = (\boldsymbol{A}-\boldsymbol{D}) + \boldsymbol{D} \tag{4.37}$$

将其代入式（4.36），得到

$$\boldsymbol{D}\,\boldsymbol{x} = \boldsymbol{b} - (\boldsymbol{A}-\boldsymbol{D})\boldsymbol{x} \tag{4.38}$$

由此可以求得 $\boldsymbol{x}$ 矢量：

$$\boldsymbol{x} = \boldsymbol{D}^{-1}\boldsymbol{b} - \boldsymbol{D}^{-1}(\boldsymbol{A}-\boldsymbol{D})\boldsymbol{x} \tag{4.39}$$

$$\boldsymbol{x} = \boldsymbol{D}^{-1}\boldsymbol{b} - (\boldsymbol{D}^{-1}\boldsymbol{A}-\boldsymbol{I})\boldsymbol{x} \tag{4.40}$$

式（4.39）就是雅可比法计算 x 值迭代序列的一种简化的公式。实际计算时所用的公式为

$$\boldsymbol{x}^{(k)} = \boldsymbol{D}^{-1}\boldsymbol{b} - (\boldsymbol{D}^{-1}\boldsymbol{A}-\boldsymbol{I})\boldsymbol{x}^{(k-1)} \tag{4.41}$$

下面的例 4.3 就用了这个公式。重复执行这步计算，直到满足所要求的允许误差为止。

例 4.3　利用雅可比迭代法求解以下 $\boldsymbol{Ax}=\boldsymbol{b}$ 方程组：

$$\begin{bmatrix} 3 & 1 & -1 \\ 1 & 6 & 5 \\ 1 & 1 & 3 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 7 \\ 0 \\ 5 \end{bmatrix}$$

解：

MATLAB 程序为

```
% example4 _3 - This program solves the system Ax = b using
% the function JACOBI.M  The Jacobi method is iterative, hence
```

```
% guess estimates for x0 and a tolerance level should be provided.

clc; clear all;

n = input('Number of equations = ');

% Matrix of coefficients (Input A based on your problem)
A = ...
    [3, 1, -1
    1, 6, 5
    1, 1, 3];
% Vector of constants (Input b based on your problem formulation)
b = [7; 0 ; 5];
tol = input('Convergence criterion = '); % Default tolerance is 1e-6
% Make sure you input a vector here
guess = input('Vector of initial guesses = ');

% Solution
X = Jacobi(A,b,guess,tol,1);

for k = 1:n
     fprintf('x(% 2d) = % 6.4g\n',k,X(k))
end
```

以下 MATLAB 函数完成雅可比法:

```
function x = Jacobi(A, c, x0, tol, trace)
% JACOBI Solves a set of linear algebraic equations by the
%   Jacobi iterative method.
%
%   JACOBI(A,C,X0) finds unknowns of a set of linear algebraic
%   equations. A is the matrix of coefficients, C is the vector
%   of constants and X0 is the vector of initial guesses.
%
%   JACOBI(A,C,X0,TOL,TRACE) finds unknowns of a set of linear
%   algebraic equations and uses TOL as the convergence test.
```

```
%   A nonzero value for TRACE results in showing calculated
%   unknowns at the end of each iteration.
%
%   See also GAUSS, JORDAN

% (c) N. Mostoufi & A. Constantinides
% January 1, 1999

% Initialization
if nargin<4 |isempty(tol)
    tol=1e-6;
end
if nargin>=4 & tol==0
    tol=1e-6;
end
if nargin<5 |isempty(trace)
    trace=0;
end
if trace
    fprintf('\n Initial guess :\n')
    fprintf('% 8.6g ',x0)
end

c=(c(:).')';     % Make sure it's a column vector
x0=(x0(:).')';       % Make sure it's a column vector

n=length(c);
[nr nc]=size(A);

% Check coefficient matrix, vector of constants and
% vector of unknowns
if nr~=nc
    error('Coefficient matrix is not square.')
end
if nr~=n
```

```
    error('Coefficient matrix and vector of constants do not have
the same length.')
end
if length(x0)~=n
    error('Vector of unknowns and vector of constants do not have
the same length.')
end

% Check if the coefficient matrix is singular
if det(A)==0
    fprintf('\n Rank=% 7.3g\n',rank(A))
    error('The coefficient matrix is singular.')
end

% Building modified coefficient matrix and modified
% vector of coefficients
D=diag(diag(A));                % The diagonal matrix
a0=inv(D)*A-eye(n);             % Modified matrix of coefficients
c0=inv(D)*c;                    % Modified vector of constants

x=x0;
x0=x+2* tol;
iter=0;

%  Substitution procedure
while max(abs(x-x0))>=tol
    x0=x;
    x=c0-a0 * x0;
    if trace
        iter=iter+1;
        fprintf('\n Iteration no. % 3d\n',iter)
        fprintf('% 8.6g ',x)
    end
end
```

程序运行之后输出的结果为

```
Number of equations = 3
Convergence criterion = 1e - 2
Vector of initial guesses = [1;1;1]

  Initial Guess    :
       1          1          1
  Iteration no.    1
  2.33333        -1          1
  Iteration no.    2
       3   -1.22222   1.22222
  Iteration no.    3
  3.14815   -1.51852   1.07407
  Iteration no.    4
  3.19753   -1.41975   1.12346
  Iteration no.    5
  3.18107   -1.46914   1.07407
  Iteration no.    6
  3.18107   -1.42524   1.09602
  Iteration no.    7
  3.17375   -1.44353   1.08139
  Iteration no.    8
  3.17497   -1.43012   1.08993
  Iteration no.    9
  3.17335   -1.43743   1.08505
x(1) =3.173
x(2) = -1.437
x(3) =1.085
```

4.5.2　高斯-赛德尔迭代法

高斯-赛德尔（Gauss-Seidel）迭代法与上述雅可比迭代法的区别是，每个新的迭代值一旦求得之后，立刻替代原来旧的值，用于接下来的计算，因此，不需要分别保存旧的和新的两组变量值。对于上述 3×3 矩阵，高斯-赛德尔迭代公式为

$$
\begin{aligned}
x_1^{(k)} &= -\frac{a_{12}}{a_{11}}x_2^{(k-1)} - \frac{a_{13}}{a_{11}}x_3^{(k-1)} + \frac{b_1}{a_{11}} \\
x_2^{(k)} &= -\frac{a_{21}}{a_{22}}x_1^{(k-1)} - \frac{a_{23}}{a_{22}}x_3^{(k-1)} + \frac{b_2}{a_{22}} \\
x_3^{(k)} &= -\frac{a_{31}}{a_{33}}x_1^{(k)} - \frac{a_{32}}{a_{33}}x_2^{(k)} + \frac{b_3}{a_{33}}
\end{aligned} \tag{4.42}
$$

于是，高斯-赛德尔法的算法为

$$
x_i^{(k+1)} = \frac{1}{A_{i,i}}\left(b_i - \sum_{j=1}^{i-1} A_{i,j}x_j^{(k+1)} - \sum_{j=i+1}^{N} A_{i,j}x_j^{(k)}\right) \tag{4.43}
$$

求解线性方程组还有一种方法称为 LU 分解法，详见附录 C。利用 LU 分解，上述方程组（4.43）可以重新写成如下形式：

$$
(\boldsymbol{D} + \boldsymbol{L})\boldsymbol{x}^{(k+1)} = -\boldsymbol{U}\,\boldsymbol{x}^{(k)} + \boldsymbol{b} \tag{4.44}
$$

式中 $\boldsymbol{D}$——对角矩阵；

$\boldsymbol{L}$——对角线元素为 0 的下三角矩阵；

$\boldsymbol{U}$——对角线元素为 0 的上三角矩阵。

这 3 个矩阵分别包含矩阵 $\boldsymbol{A}$ 的对角线元素、严格下三角元素和严格上三角元素。

例 4.4 用高斯-赛德尔迭代法求解如下对角占优矩阵的方程组

$$
\begin{bmatrix} 3 & 1 & -1 \\ 1 & 6 & 5 \\ 1 & 1 & 3 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 7 \\ 0 \\ 5 \end{bmatrix}
$$

解：

MATLAB 程序为

```
% example4 _ 4. m-This program solves the system Ax = b using the
% function GSEIDEL. M. The Gauss-Seidel method is iterative, hence
% guess estimates for x0 and a tolerance level should be provided.
clc; clear all;
n = input ('Number of equations = ');
% Matrix of coefficients (Input A based on your problem)
A =...
   [3, 1, -1
   1, 6, 5
   1, 1, 3];
% Vector of constants (Input b based on your problem formulation)
```

```
b=[7; 0 ; 5];
tol=input('Convergence criterion='); % Default tolerance is 1e-6
% Make sure you input a vector here
guess=input('Vector of initial guesses=');
% Solution
X=GSeidel(A,b,guess,tol);
disp(' '); disp(' ')
for k=1:n
    fprintf('x(% 2d)=% 6.4g\n',k,X(k))
end

function x=GSeidel(A, b, x0, tol)
%   GSeidel Solves a set of linear algebraic equations by the
%   the Gauss Seidel iterative method.
%
%   GSeidel(A,b,X0, TOL) finds unknowns of a set of linear
%   algebraic equations. A is the matrix of coefficients,
%   b is the vector of constants, X0 is the vector of initial
%   guesses, and TOL is the convergence criterion.
% Initialization
fprintf('\n Initial guess :\n')
fprintf('% 8.6g ',x0)
b=(b(:).')';   % Define the vector of coefficients
x0=(x0(:).')'; % Define the vector of guess estimates
d=diag(A);     % Extracting the diagonal elements of A
Q=A-diag(d);      % Defining the iteration matrix
n=length(b);
[N N]=size(A);
% Check if the coefficient matrix is singular
if det(A)==0
    fprintf('\n Rank=% 7.3g\n',rank(A))
    error('The coefficient matrix is singular.')
end
% Building modified coefficient matrix and modified
% vector of coefficients
```

```
x = x0;
x0 = x + 2 * tol;
iter = 0;
% Substitution procedure
while max(abs(x - x0)) > = tol
     x0 = x;
     for i = 1:N
           x(i) = (b(i) - Q(i,:) * x)/d(i);
    end
    iter = iter + 1;
    fprintf('\n Iteration no. % 3d\n',iter)
    fprintf('% 8.6g ',x)
end
```

程序运行之后的输出结果为

```
Number of equations = 3
Convergence criterion = 1e - 2
Vector of initial guesses = [1;1;1]

  Initial guess :
        1          1          1
  Iteration no.    1
  2.33333   -1.22222     1.2963
  Iteration no.    2
  3.17284   -1.60905     1.1454
  Iteration no.    3
  3.25149   -1.49642    1.08164
  Iteration no.    4
  3.19269   -1.43348    1.08027
  Iteration no.    5
  3.17125   -1.42876    1.08584
  Iteration no.    6
  3.17153   -1.43345    1.08731

 x(1) = 3.172
```

```
x(2) = -1.433
x(3) =1.087
```

通常，如果雅可比迭代法收敛，那么，高斯-赛德尔迭代法也收敛，并且其收敛速度更快，一般是雅可比法收敛速度的2倍。应用雅可比法和高斯-赛德尔法时，一般要求矩阵 $\boldsymbol{A}$ 为对角占优矩阵。矩阵 $\boldsymbol{A}$ 为对称正定矩阵是连续迭代收敛的一个充分条件，但不是必要条件（参见附录C）。

本章介绍了4种数值近似解的求解方法，建议读者练习这些方法的使用，并用MATLAB自带的内置函数验证所得到的答案。MATLAB函数是求解矩阵方程的有效方法，应该尽可能使用，特别是对于低阶非奇异矩阵。例如，在指令窗中键入指令 `x=A^-1*b`，就可以方便地求矩阵 $\boldsymbol{A}$ 的逆，并把 $\boldsymbol{A}$ 的逆矩阵乘以矢量 $\boldsymbol{b}$，直接得到 $\boldsymbol{x}$ 的估计值。

现在，就用MATLAB内置函数重新求解例4.3和例4.4中的方程组。

```
>> A=[3,1,-1;1,6,5;1,1,3]
A=
     3    1     -1
     1    6      5
     1    1      3
>> b=[7; 0; 5]
ans=
     7
     0
     5
>> x=A^-1*b
x=
    3.1739
   -1.4348
    1.0870
```

4.6　本章学习要点

学完本章之后，读者应该掌握以下内容：

1）生物医学领域的线性代数系统可以用数值公式求解。

2）矩阵代数是求解线性代数方程组最重要的工具（参见附录C）。

3）MATLAB 有完成矩阵运算的函数指令。

4）求解方程组 $\boldsymbol{Ax}=\boldsymbol{b}$ 的方法有迭代法和非迭代法等。

5）正确选择适当的数值方法求解线性代数方程组。

4.7 习题

4.1 请编写一个程序，利用高斯消元法将以下系数矩阵［$\boldsymbol{A}$］进行 LU 分解，求出 $\boldsymbol{L}$ 矩阵和 $\boldsymbol{U}$ 矩阵，并验证［$\boldsymbol{A}$］＝［$\boldsymbol{L}$］［$\boldsymbol{U}$］。然后用以下 MATLAB 矩阵运算函数检验答案的正确性：［$\boldsymbol{L}$，$\boldsymbol{U}$］＝lu（$\boldsymbol{A}$）。其中，$\boldsymbol{A}$＝［1 3 6；4 5 8；1 2 3］。

4.2 代谢工程中的线性方程

微生物能够单独利用葡萄糖、甲醇或者十六烷作为底物进行繁殖。按照重量分析，已知某微生物细胞的平均成分为碳占 47%、水占 6.5%、氧占 31%、氮占 10%，其余为灰粉。在活性新陈代谢过程中，微生物将底物、氧和氮转化成生物物质、二氧化碳和水。通过分析微生物培养过程中底物和氧气的消耗量可以估计细胞的产量。微生物培养时，向生物反应器内持续通入空气，产生的废气则通过反应器顶端的通气口排出。并且，用质谱仪分析进气和出气的成分。分析之前，要去除进气和出气中所含的水气。进气就是环境空气（21% 氧和 79% 氮）。出气中各种成分所占的体积百分比如下。请根据氮和氧的变化计算由消耗底物葡萄糖而生成的微生物细胞的产率。

底物	氮%	二氧化碳%	氧%
葡萄糖	78.8	10.2	11.0

4.3 光谱仪在生物溶液测定中的应用

光谱仪使用最多的是测定溶液中生物分子的浓度。假设在波长 i 处溶液的总吸光度为 A_i，则 A_i 等于溶液中各个成分 k（$k=1\sim N$）的吸光度之和，而每种溶液成分 k 在该波长下的吸光度等于该成分单位长度消光系数 $\varepsilon(k)$ 与该成分浓度 $C(k)$ 的乘积（Tinoco 等人，2002），即

$$A_i = \left(\sum_{k=1}^{N} \varepsilon_i(k)C(k)\right)$$

假设某蛋白质溶液含有 M、N、O 和 P 这 4 种氨基酸。如下表所示，实验中用 4 个不同波长的光测定各种氨基酸的浓度，表中列出了各个波长下 4 种氨基酸的消光系数以及实验测得的吸光度值。请分别用高斯-若尔当法和雅可比迭代法计算这 4 种氨基酸的浓度（单位为 M），并比较两种算法的结果和效率。

波长/nm	ε_M（M^{-1}）	ε_N（M^{-1}）	ε_o（M^{-1}）	ε_p（M^{-1}）	A_i
240	11300	8150	4500	4000	0.6320
250	5000	7500	3650	4200	0.5345
260	1900	3900	3000	4800	0.3310
280	1500	1400	2000	4850	0.1960

4.4　生物制药中药物的三相萃取

试考虑从有机相萃取某种药物到两个水相的问题。两个有机相的体积流率分别为 H 和 K，两个水相的体积流率分别为 L 和 M。如下表所示，用不同的 H、K、L 和 M 值进行了 4 次不同的实验，不同实验中 4 个液流的药物出口浓度相同，都分别为 y、w、x 和 z。每次实验的溶液药物总量表示为 F。请计算 4 个液流的药物出口浓度 y、w、x 和 z。

实验序号	H/（L/min）	K/（L/min）	L/（L/min）	M/（L/min）	总量 F/g
1	100	125	125	62.5	6625
2	80	110	120	25	5290
3	140	80	120	100	7300
4	90	104.8	60	137.33	6539

药物在 4 个液流中的分布与每次实验的溶液药物总量之间遵循如下关系：

$$Hy + Kw + Lx + Mz = F$$

4.8　参考文献

Constantinides, A. and Mostoufi, N. 1999. *Numerical Methods for Chemical Engineers with MATLAB Applications.* Upper Saddler River, NJ: Prentice Hall PTR.

Koehler, J. K. 1973. *Advanced Techniques in Biological Electron Microscopy.* New York: Springer Verlag.

Rosenfeld, A. and Kak, A. C. 1982. *Digital Picture Processing*, 2nd ed. New York: Academic Press.

Shuler, M. L. and Kargi, F. 2002. *Bioprocess Engineering: Basic Concepts*, 2nd ed. Englewood Cliffs, NJ: Prentice Hall.

Tinoco, I., Sauer, K., Wang, J. C., and Puglisi, J. D. 2002. *Physical Chemistry: Principles and Applications in Biological Sciences*, Upper Saddle River, NJ: Prentice Hall.

第 5 章　生物医学系统中的非线性模型

5.1　绪论

生物医学工程中有很多问题需要求解非线性方程，并且，这与线性方程不同，大多数非线性方程不能用解析法求解，而线性方程的根可以用解析法求得。因此，求解非线性方程的根需要用数值方法。本章介绍识别非线性方程的各种方法，并将着重讲述生物医学工程中各个核心研究领域中的一些非线性方程的例子，用于说明数值求解方法的应用。

本章包括如下学习内容：

1）识别生物医学工程模型中的非线性方程；

2）了解非线性方程求根的各种不同方法（如逐次代换法、试位法、牛顿—拉弗森法等）；

3）学习非线性方程组求解的牛顿法；

4）编写 MATLAB 程序实现非线性方程求解的主要算法；

5）了解非线性方程求根方法的收敛性。

5.2　非线性方程的一般形式

非线性方程和线性方程之间很容易区分，例如，以下两个方程的变量 x 显然是非线性方程，因为不能用解析方法从这些方程中分离出 x，从而得到方程的根。

$$\ln(x) + x = 4.6; \quad \frac{1}{\sqrt{x}} = \sin(\sqrt{x} + 0.01)$$

包含 $\log(x)$ 、$\sqrt{x}$、$\sin(x)$ 等项的方程肯定是非线性的。一般，如果方程中含有变量的非线性函数、或者变量的高次项，比如，x^2、x^3、$\log(x)$ 、$\sin(x)$ 、$\exp(x)$ 等，那么该方程为非线性方程。多项式方程也是非线性方程，可以表示为

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots\cdots + a_1 x + a_0 = 0$$

多项式方程的根就是可以使 $f(x)$ 等于 0 的变量 x 的值。如果 $f(x)$ 是二次方程，则可以方便地求得解析解。三次以及三次以上的高次多项式方程，有 3 个或

3个以上的根，利用下节介绍的数值方法通常更容易求取这些方程的根。

除了方程根的数目之外，还要重视根的性质，特别是生物医学工程问题中出现的高次多项式方程。这些方程的根可以是：（a）不同的实根；（b）重复的实根；（c）复数根；（d）a～c各种根的组合。根的实数部分可以是正数、负数或0。

下面来看两个三次多项式方程的例子。第一个多项式为

$$x^3 - 7x + 6 = 0$$

如图5.1a所示，用如下MATLAB指令作出函数曲线，从图上可以看出方程的根：

```
x = linspace(-4,3,30);          % 定义 x 的取值范围
y = x.^3 - 7 * x + 6;           % 计算函数值
xx = linspace(0,0,30);          % 定义直线 y = 0
plot(x,y, x,xx)                 % y 函数作图
xlabel('x'); ylabel('y');       % 给两个坐标轴加上标题
```

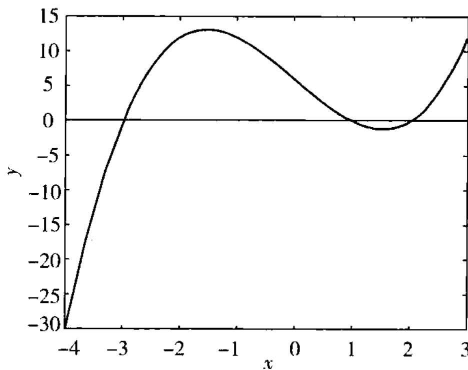


图5.1a　函数曲线与 $y=0$ 的3个交点就是3个实根

第二个多项式为

$$x^3 - 2x^2 + x - 2 = 0$$

同样，用MATLAB指令作出函数曲线（见图5.1b），也可以看出方程的根。

第一个方程的根为2、-3和1，是3个实根。第二个方程只有1个实根，即 $x=2$，以及2个共轭复数根，$x=-\mathrm{i}$ 和 $x=\mathrm{i}$。注意，只有实根才能由图上函数曲线与直线 $y=0$ 的交点显示出来。

利用MATLAB函数fzero，可以找出任意一元方程的局部根。使用这条指令时，需要先建立一个独立的函数文件，例如，取名为file_name.m，则指令fzero（'file_name'，x_0）就是找出最接近估计值 x_0 并使该函数等于0的局部根。

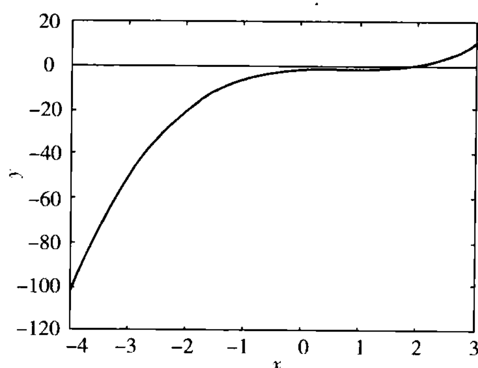


图 5.1b 函数曲线与 $y=0$ 只有 1 个交点，即 1 个实根，其他 2 个共轭复数根在图上显示不出来

输入不同的初始估计值 x_0，每次运行这条指令可以找出方程的一个根。下面是用 fzero 指令求解上述第二个方程的过程：

先建立函数文件 function2. m：

```
function y = function2(x)
y = x^3 - 2 * x^2 + x - 2;
```

然后，在 MATLAB 指令窗中运行如下指令：

```
>> x = linspace(-4,3,30);
>> x = fzero('function2',sqrt(-1))
x = 0 + 1.0000i
>> x = fzero('function2', - sqrt(-1))
x = 0 - 1.0000i
>> x = fzero('function2',4)
x = 2
```

就得到了方程的 3 个解。

5.3 非线性生物医学系统举例

下面按照生物医学工程的不同研究领域，举例介绍几个生物医学工程中的非线性方程。

5.3.1 分子生物工程

作为生化反应中的催化剂，酶是一种细胞衍生的蛋白质。酶对底物具有非常高的特异性。底物的消耗量可用酶活性底物消耗的动力学过程来描述。酶促反应的动力学过程可以用如下 Michaelis-Menten 模型方程表示：

$$r_s = -\frac{\mathrm{d}s}{\mathrm{d}t} = \frac{V_{\max}s}{K_m + s}$$

式中　r_s——底物的反应速率，单位为摩尔/体积/时间；

s——底物的浓度，单位为摩尔/体积；

$V_{\max}$——最大底物消耗率；

K_m——酶促反应速率达到最大反应速率一半时的底物浓度。

重排这个方程，把浓度 s 和时间 t 这两个变量分开，就可以用积分法求得底物浓度随时间变化的非线性解：

$$K_m \ln\left(\frac{s_0}{s}\right) + (s_0 - s) = V_{\max}t$$

式中　s——t 时刻底物的浓度；

s_0——底物的初始浓度。

要知道在某种给定酶的作用下，某时刻的底物浓度 s，就必须用数值方法求解这个方程。图 5.2 所示为底物浓度 s 的一种典型消耗曲线。本章后面将用牛顿—拉弗森法求解这个方程（见例 5.3）。

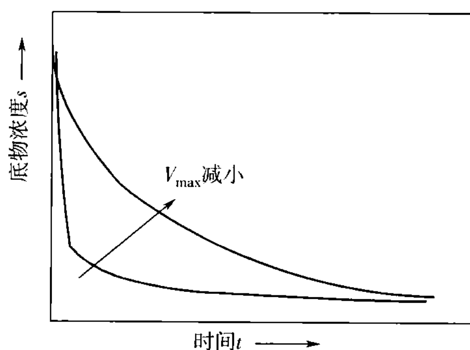


图 5.2　由酶催化的生化反应中底物浓度的变化是非线性的，需要用本章介绍的非解析求解方法求取任意时间 t 的底物浓度 s

5.3.2 细胞和组织工程

哺乳动物的细胞在胞外组织基质中的运动是细胞和组织工程的一个重要研究课题，组织工程疗法中的创伤修复、组织再生以及植入支架的再生诱导等都必须

有细胞迁移。例如：微循环系统血管壁上的内皮细胞需要迁移到组织中才能生成新的血管网络，这个过程称为血管再生；白血球要沿着人造血管材料移动，以便到达目标位点并消灭细菌微生物，这个过程就是急性炎症反应中出现的现象。如果用计算机辅助慢速摄影数字显微镜直接拍摄细胞运动的位置和轨迹，就可以用如下 Dunn 方程定量描述细胞的群体迁移：

$$< d^2 > = 2S^2[Pt - P^2(1 - e^{-t/P})]$$

如图 5.3 所示，细胞位移平方的均值 $< d^2 >$ 为时间 t 的非线性函数，方程的自变量是细胞运动速度的均方根 S 以及持续时间 P。用数值回归方法可以确定这两个参数的值。如果用独立的方法已经得到其中的一个值，则另一个值可以用数值方法确定。

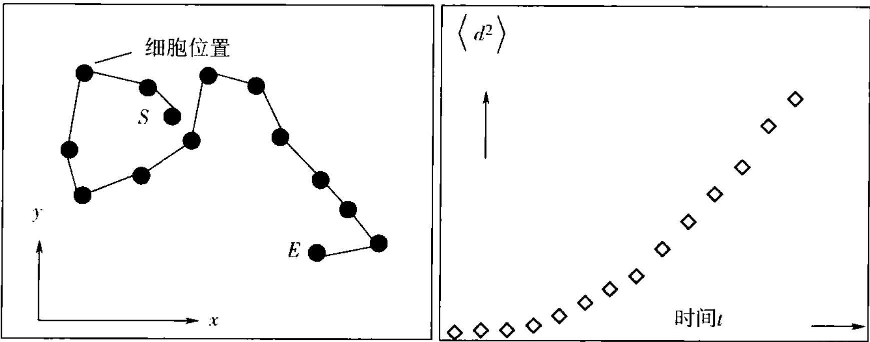


图 5.3　左图为单细胞迁移的二维轨迹图（S 表示起始位置，E 表示终止位置），右图为细胞位移平方的群体均值 $< d^2 >$ 与时间 t 的关系

利用本章所述的迭代法求 Dunn 方程的数值解，就可以得到单细胞的运动速度和持续时间。

5.3.3　生物热传导——光热疗法

光诱导的组织加热有各种各样的应用，可以作为神经生长和创口愈合的生物刺激，用于外科手术中的血管封闭和焊接、癌症和前列腺增生治疗中的组织破坏，以及激光角膜切除（Enderle 等人，2005）视力矫正手术中的组织消融等。激光疗法将光子能量转化为生物组织可吸收的能量。如图 5.4 所示，在激光照射过程中，不同时间和不同空间深度下组织表面温度的变化情况由偏微分方程数学模型描述。图中 θ 为无量纲温度，ξ 为无量纲空间深度，τ 为组织消融发生之前的光照时间（无量纲）。

假设 $\xi = 0$ 处 $\theta = 1$，并设 λ 为消融的阈值温度，B 为无量纲光吸收参数，则如下超越代数方程描述了这些量之间的关系：

$$B\lambda = \frac{2}{\sqrt{\pi}} = B\sqrt{\tau_{ab}} + e^{B^2\tau_{ab}}erfc[B\sqrt{\tau_{ab}}] - 1$$

必须用数值方法求解这个方程，才能得到图 5.4 所示的诱发产生组织消融所需温度 λ 随着光照时间 τ_{ab} 变化的曲线。

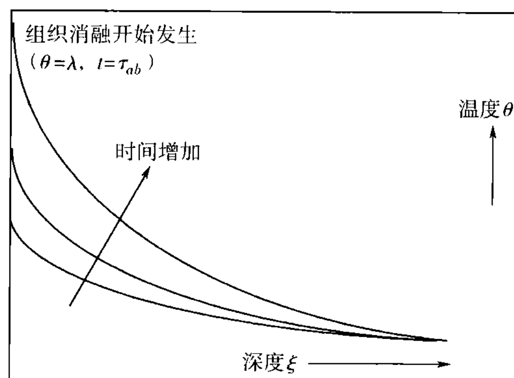


图 5.4　在消融发生之前不同的光照时间下，无量纲表面温度随组织深度变化的曲线（表面温度由上述非线性方程的数值解得到）

5.3.4　生物医学中的流体传输动力学

生物流体传输与许多生理过程相关，例如：血液的对流传导，氧分子在浓度差驱动下向周围组织的扩散传输，以及离子在离子泵和电化学梯度作用下的进出细胞的输运过程等。病人治疗期间，如手术前或手术中，还会需要输液。假设某器官中输送血液的血管直径 D 为 2.8mm，血液流速为 $u = 1.8\text{m/s}$，密度为 $\rho = 1000\text{kg/m}^3$，粘度为 $\mu = 0.001\text{kg/(m·s)}$。通常，血管中流动的液体形成层流，但是，如果血管某处流速过高，也可能发生湍流。惯性力与粘性力之比是一个无量纲数，称为雷诺（Reynolds，Re）数，$\text{Re} = Du\rho/\mu$。当 Re 的值大于经验阈值 2000 时，可以认为流体呈现湍流，此时，可以用摩擦系数 b 确定液体流过管道时的速度变化，而 b 由如下非线性经验方程给出，该方程称为 Colebrook 方程。显然，必须用数值方法求解该方程才能得到 b 值。

$$\frac{1}{\sqrt{b}} = 4.07\ln(\text{Re}\sqrt{b}) - 0.60$$

5.4　逐次代换法

下面介绍单点迭代搜索根的最简单方法。先变换方程 $f(x) = 0$，使方程的左边只有一个 x：

$$x = g(x) \tag{5.1}$$

函数 $g(x)$ 就是用于估算根的公式。直线 $y=x$ 与曲线 $y = g(x)$ 的交点就是方程的根。如图 5.5a 所示，选取一个初始近似值 x_1，可以求得 x_2：

$$x_2 = g(x_1) \tag{5.2}$$

再把 x_2 作为新的初始值，进行下一个迭代。以此类推，这种迭代法的计算公式为

$$x_{n+1} = g(x_n) \tag{5.3}$$

这种方法称为逐次代换法，$x = g(x)$ 就是代换公式。

该方法的收敛条件是对于搜索区间上所有的 x 值，必须满足 $|g'(x)| < 1$。如图 5.5b 所示，如果不满足这个条件，则搜索结果是发散的。在计算机的程序中可以检查是否满足条件 $|x_3 - x_2| < |x_2 - x_1|$，如果这种关系成立，则 x_n 值的序列是收敛的。逐次代换法的优点是只需要一个出发点，并且不需要求函数的导数。

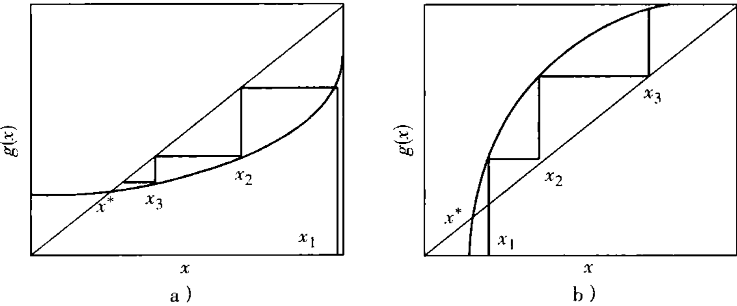


图 5.5 逐次代换法

a) 收敛 b) 发散

5.5 试位法（线性插值法）

这种寻找根的方法基于函数中两点之间的线性插值，选取的这两个点之间要包含根，也就是这两个点位于根的两边。例如，图 5.6a 中的 x_1 和 x_2 位于非线性函数 $f(x)$ 的根 x^* 的两边。插值就是用直线段将 $(x_1, f(x_1))$ 和 $(x_2, f(x_2))$ 两点连起来，该线段称为弦，其方程为

$$y(x) = mx + c \tag{5.4}$$

因为该弦通过 $(x_1, f(x_1))$ 和 $(x_2, f(x_2))$ 两个点，其斜率即为

$$m = \frac{f(x_2) - f(x_1)}{x_2 - x_1} \tag{5.5}$$

其 y 截距为

$$c = f(x_1) - mx_1 \tag{5.6}$$

式（5.4）就成为

$$y(x) = \left[\frac{f(x_2) - f(x_1)}{x_2 - x_1}\right]x + \left\{f(x_1) - \left[\frac{f(x_2) - f(x_1)}{x_2 - x_1}\right]x_1\right\} \tag{5.7}$$

设 $y(x_3) = 0$，就可以求得下一个更接近根的估计值 x_3。将 $x = x_3$ 以及 $y(x_3) = 0$ 代入式（5.7），有：

$$x_3 = x_1 - \frac{f(x_1)(x_2 - x_1)}{f(x_2) - f(x_1)} \tag{5.8}$$

注意，对于图 5.6 所示的曲线，x_3 比 x_2（或 x_1）更接近根。但是，并不是所有函数曲线都会这样。

根据图 5.6a，$f(x_3)$ 与 $f(x_2)$ 具有相同的符号，因此，可以用 x_3 代替 x_2。再重复以上操作步骤，如图 5.6b 所示，用新的弦把 $(x_1, f(x_1))$ 和 $(x_3, f(x_3))$ 这两点连起来，就得到了第二步线性插值的值 x_4：

$$x_4 = x_1 - \frac{f(x_1)(x_3 - x_1)}{f(x_3) - f(x_1)} \tag{5.9}$$

x_4 比 x_3 更接近根。

一般，假设有 x^+ 使 $f(x^+) > 0$，而 x^- 使 $f(x^-) < 0$，那么，下一个更接近方程根的近似值可以用如下通用公式计算：

$$x_n = x^+ - \frac{f(x^+)(x^+ - x^-)}{f(x^+) - f(x^-)} \tag{5.10}$$

求下一个代换值 x_n 时，根据 $f(x_n)$ 的符号，用 x_n 替换 x^+ 或者 x^-。

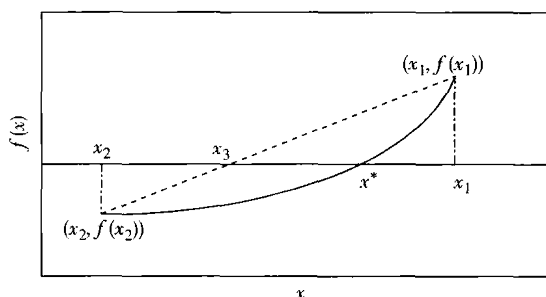


图 5.6a　线性插值的第一步

这种方法有好几个名称：弦截法、线性插值法、试位法等，其计算很简单，每一步不需要求函数的导数，因此，是求解非线性方程最方便的方法之一。但是，它的主要缺点是搜索根时，收敛的准确度和速度受到初定值 x_1 的限制，x_1 成为所有后续代换的关键点。

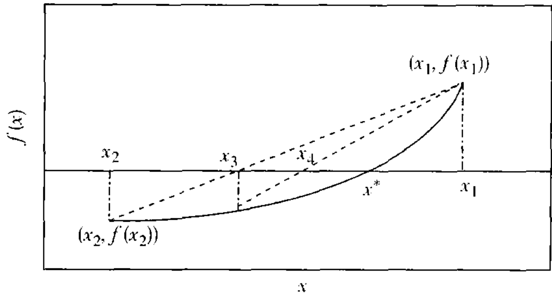


图 5.6b 线性插值的第二步

5.6 牛顿—拉弗森法

非线性方程求根的主要方法是牛顿—拉弗森法（Newton-Raphson，N-R 法）。N-R 法的第一步是把非线性方程表示为

$$f(x) = 0 \tag{5.11}$$

例如，5.3.4 节介绍的 Colebrook 方程就可以表示为

$$\frac{1}{\sqrt{b}} - 4.07\ln(\mathrm{Re}\sqrt{b}) + 0.60 = 0$$

构建好这样的非线性方程之后，就可以应用 N-R 法求解。

这里，先了解一下 N-R 法的物理意义。开始时，我们不知道方程的根 x^* 的值，但是，可以先找一些根的粗略估计值，再逐步细化，直到得到满意的值为止。N-R 法就是这样一个处理过程。

利用如下基本泰勒级数展开式，可以将非线性函数 $f(x)$ 围绕任意一点 x_1 展开，x_1 称为根的初始估计值：

$$f(x) = f(x_1) + f'(x_1)(x - x_1) + \frac{f''(x_1)(x - x_1)^2}{2!} + \cdots \tag{5.12}$$

求方程的根是找出使函数 $f(x)$ 为 0 的值 x，也就是使上式左边为 0，再求解该方程。但是，此时方程的右边是一个无穷级数，必须截断该级数序列，保留有限项。为了简单，我们只取前两项，即函数值及其变化率（也就是一阶导数）这两项，把函数 $f(x)$ 线性化，得到

$$f(x) \approx f(x_1) + f'(x_1)(x - x_1) \tag{5.13}$$

设函数的根为 $x = x^*$，将 x^* 代入上式，可得

$$f(x^*) = f(x_1) + f'(x_1)(x^* - x_1) \tag{5.14}$$

因为 x^* 为 $f(x) = 0$ 的根，所以有 $f(x^*) = 0$，重排此式，可得 x^* 的表达式

$$x^* = x_1 - \frac{f(x_1)}{f'(x_1)} \tag{5.15}$$

该式表示的物理意义是：函数 $f(x) = 0$ 的根 x^* 可以由初始估计值 x_1 减去 $f(x_1)/f'(x_1)$ 得到。图 5.7a 表示了这种含义，读者应该充分理解其含义。此图表示，沿着函数 $f(x)$ 的切线 $f'(x_1)$ 的方向，将 x_1 向 x^* 移动，可以求取根 x^*。由 $(x_1, 0)$、$(x^*, 0)$ 和 $(x_1, f(x_1))$ 3 个顶点组成的直角三角形，其斜边的斜率为 $f'(x_1) = f(x_1)/(x_1 - x^*)$。

但是，实际上，我们不可能一步就到达 x^* 点，因为我们只保留了两项泰勒级数，从式（5.15）得到的 x^* 值并不满足 $f(x) = 0$，如图 5.7a 所示，第一次应用式（5.15）只不过产生了图中箭头所指的根的一个新估计值，可以称为 x_2，即

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)} \tag{5.16}$$

如图 5.7b 所示，N-R 法重新利用泰勒级数，把 x_2 处的函数 $f(x)$ 线性化，得到根 x^* 更好的一个估计值，称为 x_3，即

$$x_3 = x_2 - \frac{f(x_2)}{f'(x_2)} \tag{5.17}$$

继续重复上述过程，直至最后收敛到根 x^* 为止。因此，N-R 法第 n 次计算的迭代公式为

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{5.18}$$

此式的另一种表示形式是用根的新估计值 x_{new} 和原估计值 x_{old} 来表示。读者会发现，在编写数值计算程序实现这种迭代过程时，这种表示方式很有用，前一次迭代的 x_{new} 成为下一次迭代的 x_{old}，如此重复：

$$x_{new} = x_{old} - \frac{f(x_{old})}{f'(x_{old})} \tag{5.19}$$

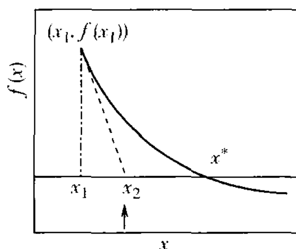


图 5.7a

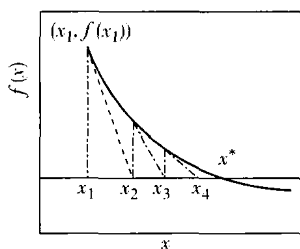


图 5.7b

总之，N-R 法每次都用新求得的估计点作为下一次迭代的起点。

图 5.7a 和 b 显示了经过逐次迭代，最终使搜索收敛于根的过程。但是，非线性函数的变化很大，并不是所有的搜索都能收敛。如图 5.8a 所示，实际上经常会产生发散现象，要非常小心地选择初始起点才能保证收敛（见图 5.8b）。

为了研究 N-R 法的收敛性，需要考察式（5.18）的 $-f(x_n)/f'(x_n)$ 项，这是误差项，也称校正项，用于校正前一个根的估计值（即第 n 次迭代的估计值）。如果在根的附近函数轨迹很陡，那么函数的一阶导数（即斜率）就很大，也就是误差项的分母很大，因此，估计值的收敛就很快。但是，如果在根的附近函数 $f(x)$ 很平坦，则收敛就会很慢。那么，搜索在什么点上，N-R 法不会收敛呢？根据式（5.18），可以推测，如果分母 $f'(x_n)$ 为0，比值 $f(x_n)/f'(x_n)$ 就不能确定。确实，搜索范围内曲线上存在的回折点是件麻烦事，可能引起搜索的发散。

N-R 法收敛的一个条件是："在区间 (x_1, x^*) 上，如果 $f'(x)$ 和 $f''(x)$ 的符号保持不变，也就是 $f(x)$ 的斜率和 $f'(x)$ 的斜率不存在回折点，并且，$f'(x_1)$ 和 $f''(x_1)$ 具有相同的符号，那么，迭代总是收敛于 x^*。"这条收敛准则可以作为实现 N-R 搜索算法的 MATLAB 程序的一个部分，用于判断收敛性。显然，图 5.8a 的情况不满足这条准则。

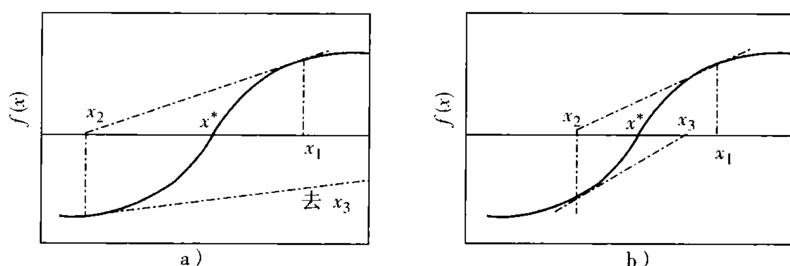


图 5.8 初始估计值的选择会影响 N-R 法的收敛性

a）发散 b）收敛

例 5.1 心血管生理系统——利用 N-R 法求解简单多项式

问题陈述：

假设左心室压力（Left Ventricular Pressure，LVP）可以用一个抛物线表示，收缩压的峰值为 120mmHg，并且平均主动脉压为 90mmHg。请利用 N-R 法计算主动脉瓣打开和关闭的时间。如果收缩压的峰值从 120mmHg 降到 100mmHg 或者升到 140mmHg，这两个时间又是多少？（提示，主动脉瓣打开和关闭的发生时刻就是 LVP 与平均主动脉压相等的时刻。）

解：

如图 5.9 所示是收缩压峰值为 120mmHg 的 LVP 抛物线，曲线与横坐标 x 有

两个交点，一个是原点（0，0），另一个则是一个心动周期的结束点（1，0），这里将一个周期的持续时间归一化为1。抛物线的方程为

$$(x-h)^2=-4p(y-k)$$

式中 (h, k)——抛物线顶点的坐标；

p——抛物线的焦点。

假设抛物线是对称的，则初始 $k=120\text{mmHg}$ 时，(h, k) 的值为（0.5，120），其中0.5为归一化周期时间1的中点。此时 p 未知，可以从抛物线与横坐标 x 的交点（0，0）或（1，0）求得 p 的值为 5.208×10^{-4}。于是，当 $k=120\text{mmHg}$ 时，抛物线的轨迹如图5.9所示，即

$$(x-0.5)^2=-4(5.208\times10^{-4})(y-120)$$

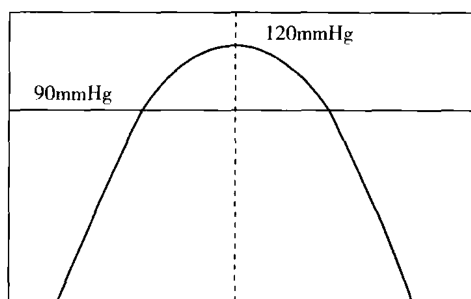


图5.9 抛物线方程的曲线

本题需要求解抛物线上左心室压力等于平均主动脉压90mmHg的点，因此，将 $y=90$ 代入抛物线方程，可得方程 $480x^2-480x+90=0$。对于不同的收缩压峰值 k，方程的通式为

$$x^2-x+\frac{22.5}{k}=0$$

这些方程要用N-R法求根。

下面是两种不同的求解方法。第一种方法是一种简化方法，直接使用函数的准确导数值，建立了一个包含抛物线函数及其导数的MATLAB文件，命名为parabola_and_derivative.m。显然，这个文件名可以根据需要改变，只要N-R法的执行程序simple_NR.m能够正确调用就可以了。

第二种方法假设函数的准确导数值未知，要用第6章将介绍的那些数值方法近似求导。这种方法的MATLAB实现程序命名为NR2.m。

下面是实现这两种方法的程序。

1. 使用解析函数导数的简化N-R法

```
% example5_1a.m
```

```
% This program calculates the roots of a parabolic
% equation using Newton-Raphson method for different
% peak pressures, denoted by k.

clc; clear all;
k = input(' Peak Pressure in mmHg, k = ');
x0 = input(' Starting value = ');

% simple _ NR('F', X0, P1, P2, ..) is a simplified Newton-Raphson
% routine that finds the root of a nonlinear function
% contained in the MATLAB file called F with arguments P1, P2,...
% The file F contains both the function and its derivative.

xnew = simple _ NR('parabola _ and _ derivative',x0,k);
fprintf('\n % 3g % g % g\n',k,x0,xnew)
```

包含解析导数的 MATLAB 函数

```
function[ f,fprime] =parabola _ and _ derivative(x,k)
% alter the function below as necessary
f =x^2 - x +22.5/k;
% modify the function derivative below as necessary
fprime =2 * x - 1;
```

简化 N-R 法的 MATLAB 函数

```
function x = simple _ NR(FunFcn,x0,varargin)
% simple _ NR finds a zero of a defined function by the Newton -
% Raphson method.
%
% simple _ NR('F',X0) finds a zero of the function described by the
% M-file F.M. that also contains the derivative of the function.
% This program differs from NR.m because it computes the
% derivative exactly, and does not numerically approximate the
% derivative. X0 is a starting guess.
%
```

```
% simple_NR('F',X0,P1,P2,...) allows for additional arguments
% which are passed to the function F(X,P1,P2,...).
% (c) S. Dunn, A. Constantinides, and P. Moghe
% August 1, 2003

% Initialization
tol=1e-6;
iter=0;
[fnk, fpr]=feval(FunFcn,x0,varargin{:});

header=' Iteration      x           f(x)';
disp(header)
fprintf('%5.0d  %13.6g %13.6g  \n',iter,[x0 fnk])
x=x0;
x0=x+1;
itermax=100;
% Main iteration loop
while abs(x-x0)>tol & iter<=itermax
    iter=iter+1;
    x0=x;
    x=x0-fnk/fpr;
    [fnk,fpr]=feval(FunFcn,x,varargin{:});
    % Show the results of calculation
    fprintf('%5.0d  %13.6g  %13.6g \n',iter,[x fnk])
end
if iter>=itermax
    disp('Warning:Maximum iterations reached.')
end
```

程序运行的输出结果（只显示了收缩压峰值 $k=120$mmHg 时的结果）

```
 peak Pressure in mmHg, k=120
  Starting value=0
  Iteration       x             f(x)
                  0           0.1875
      1        0.1875       0.0351562
```

```
        2        0.24375      0.00316406
        3       0.249924    3.81156e-05
        4           0.25    5.80764e-09
        5           0.25    1.11022e-16
    120 0 0.25
```

2. 用数值方法近似求导的 N-R 法

```
% example5_1b.m
% This program calculates the roots of a parabolic
% equation using Newton-Raphson method for different
% peak pressures, denoted by k. Derivatives are numerically
% approximated. Use example5_1a.m as an alternative.

clc; clear all;

k = input(' Peak Pressure in mmHg, k = ');
x0 = input(' Starting value = ');
% NR2('F', X0, TOL, TRACE, P1, P2, ..) is a simplified
% Newton-Raphson routine that finds zero of function F with
% arguments P1, P2, etc. TOL is convergence tolerance.
% NR2 does not graphically show the path to convergence.
xnew = NR2('parabola',x0,[],0,k);
fprintf('%3g %g %g\n',k,x0,xnew)
```

包含所需求解方程的 MATLAB 函数

```
function y = parabola(x,k)
y = x^2 - x + 22.5/k;
```

用数值方法近似求导的 N-R 法的 MATLAB 函数

```
function x = NR2(FunFcn,x0,tol,trace,varargin)
% NR2 Finds a zero of a function by the Newton-Raphson method.
%
%   NR2('F',X0) finds a zero of the function described by the
%   Simplified Newton-Raphson Program—M-file F.M. X0 is a
```

```
%   starting guess.
%
%   NR2('F',X0,TOL,TRACE) uses tolerance TOL for convergence test.
%   TRACE = 0 does not show the calculation steps
%   TRACE = 1 shows the calculation steps numerically
%
%   NR2('F',X0,TOL,TRACE,P1,P2,...) allows for additional
%   arguments which are passed to the function F(X,P1,P2,...).
%   Pass an empty matrix for TOL or TRACE to use the default
%   value.
%
%   See also FZERO, ROOTS, XGX, LI

% (c) S. Dunn, A. Constantinides, and P. Moghe
% August 1, 2003

% Initialization
if nargin < 3 | isempty(tol)
    tol = 1e - 6;
end
if nargin < 4 | isempty(trace)
    trace = 0;
end
if tol == 0
    tol = 1e - 6;
end
if (length(x0) > 1) | (~isfinite(x0))
    error('Second argument must be a finite scalar. ')
end
iter = 0;
fnk = feval(FunFcn,x0,varargin{:});
if trace
    header = ' Iteration       x              f(x)';
    disp(header)
    fprintf('% 5.0d  % 13.6g% 13.6g \n',iter, [x0 fnk])
```

```
end

x = x0;
x0 = x + 1;
itermax = 100;

% Main iteration loop
while abs(x - x0) > tol & iter <= itermax
    iter = iter + 1;
    x0 = x;

    % Set dx for differentiation
    if x ~= 0
        dx = x/100;
    else
        dx = 1/100;
    end

    % Differentiation
    a = x - dx; fa = feval(FunFcn,a,varargin{:});
    b = x + dx; fb = feval(FunFcn,b,varargin{:});
    df = (fb - fa)/(b - a);

    % Next approximation of the root
    if df == 0
        x = x0 + max(abs(dx),1.1 * tol);
    else
        x = x0 - fnk/df;
    end

    fnk = feval(FunFcn,x,varargin{:});
    % Show the results of calculation
    if trace
        fprintf('%5.0d   %13.6g   %13.6g \n',iter, [x fnk])
    end
```

```
end

if iter >= itermax
    disp('Warning:Maximum iterations reached. ')
end
```

3. 程序运行的输出结果

	初始估计值 x (0) =0	初始估计值 x (0) =1.0
收缩压峰值/mmHg	按照周期归一化的主动脉瓣打开时间	按照周期归一化的主动脉瓣关闭时间
100	0.34	0.66
120	0.25	0.75
140	0.2	0.79

注意，以上二次抛物线方程的根其实很容易用解析方法推导出来，但是，如果心室压力波形用比较复杂的非线性方程描述，那么，这里所介绍的方法就很有用。

下面所列的 3 个程序分别用 N-R 法、逐次代换法和线性插值法 3 种不同的方法求解同一个比较复杂的非线性方程，进一步介绍各种解题方法。第一个程序是用 N-R 法求解非线性方程。

例 5.2a　应用 N-R 法求解 Colebrook 非线性方程。

问题陈述：

解 Colebrook 方程，求尿液导管的摩擦系数。如下是 N-R 形式（即 $f(x)=0$）的 Colebrook 方程：

$$\frac{1}{\sqrt{b}} - 4.07\log(\mathrm{Re}\sqrt{b}) + 0.06 = 0$$

解题思路：

这里将用 N-R 法的式（5.18）反复进行迭代计算，直到前后两次连续迭代所得到的两个近似根之差小于某个默认值（如 10^{-6}）为止。并且作图显示搜索的收敛过程，以便深入了解方程解的收敛性。

程序说明：

N-R 程序命名为 NR.m，由 Constantinides 和 Mostoufi 两人在 1999 年开发完成，用于求解非线性方程的根。其中，非线性函数作为程序的参数输入，因此，该 MATLAB 程序可以用于不同的非线性方程。

```
% example5_2a.m
% This program solves a nonlinear equation using the full
% Newton-Raphson method
% It calculates the friction factor from the Colebrook equation.

clc; clear all;

disp('Calculating the friction factor from Colebrook equation')
%  Input
Re = input('\n Reynolds No.          = ');
fname = input('\n Function containing the Colebrook equation:');
% Newton-Raphson
b0 = input(' Starting value of b = ');
b = NR(fname,b0,[ ],2,Re);
fprintf('\n Friction factor, b = % 8.7f\n',b)
```

包含所需求解方程的 MATLAB 函数

```
function y = colebrook(b, Re)
% colebrook.m
% This function evaluates the value of Colebrook equation to be
% solved by the the Newton-Raphson method.
y = 1/sqrt(b)-4.07 * log(Re * sqrt(b)) + 0.6;
```

MATLAB 函数：实现 N-R 法，求函数导数的数值近似值，并作图（见图 5.10）显示根的搜索轨迹。

```
function x = NR(FunFcn,x0,tol,trace,varargin)
% NR Finds a zero of a function by the Newton-Raphson method.
%
%   NR('F',X0) finds a zero of the function described by the
%   M-file F.M. X0 is a starting guess.
%
%   NR('F',X0,TOL,TRACE) uses tolerance TOL for convergence
%   test. TRACE = 1 shows the calculation steps numerically and
%   TRACE = 2 shows the calculation steps both numerically and
```

```
%   graphically.
%
%   NR('F',X0,TOL,TRACE,P1,P2,...) allows for additional
%   arguments which are passed to the function F(X,P1,P2,...).
%   Pass an empty matrix for TOL or TRACE to use the default
%   value.
%
%   See also FZERO, ROOTS, XGX, LI

% (c) N. Mostoufi & A. Constantinides
% January 1, 1999
% Initialization
if nargin<3 | isempty(tol)
    tol=1e-6;
end
if nargin<4 | isempty(trace)
    trace=0;
end
if tol==0
    tol=1e-6;
end
if (length(x0)>1) | (~isfinite(x0))
    error('Second argument must be a finite scalar.')
end

iter=0;
fnk=feval(FunFcn,x0,varargin{:});
if trace
  header=' Iteration       x              f(x)';
  disp(header)
  fprintf('%5.0d  %13.6g %13.6g \n',iter, [x0 fnk])
  if trace==2
    xpath=[x0 x0];
    ypath=[0 fnk];
  end
```

```
end

x = x0;
x0 = x + 1;
itermax = 100;

% Main iteration loop
while abs(x-x0) > tol & iter <= itermax
  iter = iter + 1;
  x0 = x;

  % Set dx for differentiation
  if x ~= 0
    dx = x/100;
  else
    dx = 1/100;
  end

  % Differentiation
  a = x - dx; fa = feval(FunFcn,a,varargin{:});
  b = x + dx; fb = feval(FunFcn,b,varargin{:});
  df = (fb - fa)/(b - a);

  % Next approximation of the root
  if df == 0
    x = x0 + max(abs(dx),1.1 * tol);
  else
    x = x0 - fnk/df;
  end
  fnk = feval(FunFcn,x,varargin{:});
  % Show the results of calculation
  if trace
    fprintf('% 5.0d  % 13.6g % 13.6g \n',iter, [x fnk])
    if trace == 2
      xpath = [xpath x x];
```

```
      ypath=[ypath 0 fnk];
    end
  end
end

if trace==2
  % Plot the function and path to the root
  xmin=min(xpath);
  xmax=max(xpath);
  dx=xmax-xmin;
  xi=xmin-dx/10;
  xf=xmax+dx/10;
  yc=[];
  for xc=xi:(xf-xi)/99:xf
    yc=[yc feval(FunFcn,xc,varargin{:})];
  end
  xc=linspace(xi,xf,100);
  ax=linspace(0,0,100);
  plot(xc,yc,xpath,ypath,xc,ax,xpath(1),ypath(2),'*',x,fnk,'o')
  axis([xi xf min(yc) max(yc)])
  xlabel('x')
  ylabel('f(x)')
  title('Newton-Raphson:The function and path to the root (*:ini-
tial guess ; o:root)')
end

if iter>=itermax
  disp('Warning:Maximum iterations reached.')
end
```

程序运行时的输入：

```
Calculating the friction factor from Colebrook equation
  Reynolds No.          =2e5
Function containing the Colebrook equation:'colebrook'
Starting value of b=0.0001
```

程序运行的输出结果：

```
Iteration          x              f(x)
                 0.0001          69.6643
    1          0.000233872       33.3254
    2          0.000458271       13.2796
    3          0.000697928       3.56293
    4          0.000816553       0.38613
    5          0.000832694     0.00546019
    6          0.000832929   1.43852e-006
Friction factor, b=0.0008329
```

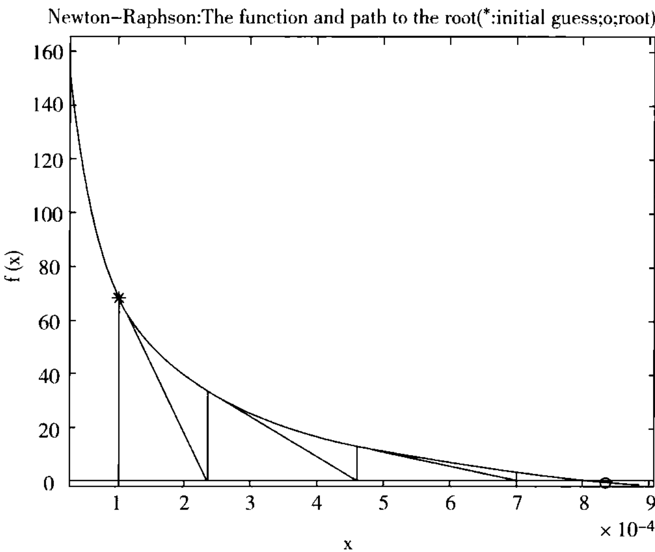


图 5.10　N-R 法的根搜索轨迹

结果讨论

有关 N-R 法求解的说明：以上的 N-R 法求解过程经过 6 次迭代就迅速收敛到方程的根。建议读者采用不同的参数测试运行这个程序，例如，把程序运行时要求输入的 Reynold 数改为 5×10^3、1×10^4、5×10^4 等，或者采用不同的初始摩擦系数估计值 x。

如果保持原来的 Reynold 数 2×10^5 不变，用不同的初始摩擦系数估计值，那么，如下表所示，N-R 法的搜索过程会发生很大的变化。请读者运行 MATLAB

程序验证这些结果。

x 的初始估计值	收敛或发散	注　释
0.001	收敛	4 次迭代
0.0015	收敛	5 次迭代
0.0017	收敛	6 次迭代；同时求得虚根
0.00175	收敛	8 次迭代；同时求得虚根
0.0018	发散	

下面的第二个程序是用逐次代换法求解同样的 Colebrook 非线性方程。

例 5.2b　应用逐次代换法求解非线性方程。

问题陈述：

解 Colebrook 方程，求尿液导管的摩擦系数。如下是逐次代换法形式（即 $x = g(x)$）的 Colebrook 方程：

$$b = \frac{1}{(4.07\log(\mathrm{Re}\sqrt{b}) - 0.60)^2}$$

解题思路：

这里用式（5.3）的逐次代换法，反复进行迭代计算，直到前后两次连续迭代得到的两个近似根之差小于某个默认值（如 10^{-6}）为止。并作图显示搜索的收敛过程，以便深入了解方程解的收敛性。

程序说明：

逐次代换法程序命名为 XGX.m，该程序用于求解非线性方程的根。其中，非线性函数作为程序的参数输入，因此，该 MATLAB 程序可以用于不同的非线性方程。

```
% example5_2b.m
% This program solves a nonlinear equation using the
% Successive Substitution method.
% It calculates the friction factor from the Colebrook equation.

clear; clc;
disp('Calculating the friction factor from Colebrook equation')
% Input
Re = input('\n Reynolds No.          = ');
fname = input('\n Function containing the Colebrook equation:');
% Successive Substitution method
b0 = input(' Starting value of b = ');
b = XGX(fname,b0,[],2,Re);
```

```
fprintf('\n Friction factor, b=% 8.7f\n',b)
```

包含被求解方程的 MATLAB 函数

```
function y=colebrookg(b, Re)
% colebrookg.m
% This function evaluates the value of Colebrook equation
% to be solved by the XGX or successive substitution method.
y=(1/(4.07*log(Re*sqrt(b))-0.6))^2;
```

MATLAB 函数：实现逐次代换法，并作图（见图5.11）显示根的搜索路径。

```
function x=XGX(FunFcn,x0,tol,trace,varargin)
% XGX Finds a zero of a function by x=g(x) method.
%
%   XGX('G',X0) finds the intersection of the curve y=g(x)
%   with the line y=x. The function g(x) is described by the
%   M-file G.M. X0 is a starting guess.
%
%   XGX('G',X0,TOL,TRACE) uses tolerance TOL for convergence
%   test. TRACE=1 shows the calculation steps numerically and
%   TRACE=2 shows the calculation steps both numerically and
%   graphically.
%
%   XGX('G',X0,TOL,TRACE,P1,P2,...) allows for additional
%   arguments which are passed to the function G(X,P1,P2,...).
%   Pass an empty matrix for TOL or TRACE to use the default
%   value.
%
%   See also FZERO, ROOTS, NR, LI

% (c) N. Mostoufi & A. Constantinides
% January 1, 1999

% Initialization
if nargin<3 | isempty(tol)
```

```
    tol =1e-6;
end
if nargin <4 | isempty(trace)
    trace =0;
end
if tol ==0
    tol =1e-6;
end
if (length(x0) >1) | (~isfinite(x0))
    error('Second argument must be a finite scalar. ')
end
if trace
  header =' Iteration       x            g(x)';
  disp(header)
  if trace ==2
      xpath =[x0];
      ypath =[0];
  end
end

x =x0;
x0 =x +1;
iter =1;
itermax =100;

% Main iteration loop
while abs(x - x0) >tol & iter <=itermax
  x0 =x;
  fnk =feval(FunFcn,x0,varargin{:});

  % Next approximation of the root
  x =fnk;

  % Show the results of calculation
  if trace
```

```
      fprintf('% 5.0f  % 13.6g % 13.6g \n',iter, [x0 fnk])
      if trace ==2
          xpath =[xpath x0 x];
          ypath =[ypath fnk x];
        end
    end
    iter =iter +1;
end

if trace ==2
    % Plot the function and path to the root
    xmin =min(xpath);
    xmax =max(xpath);
    dx =xmax -xmin;
    xi =xmin -dx/10;
    xf =xmax +dx/10;
    yc =[ ];
    for xc =xi:(xf -xi)/99:xf
        yc =[yc feval(FunFcn,xc,varargin{:})];
    end
    xc =linspace(xi,xf,100);
    plot(xc,yc,xpath,ypath,xpath(2),ypath(2),'*',...
        x,fnk,'o',[xi xf],[xi,xf],'--')
    axis([xi xf min(yc) max(yc)])
    xlabel('x', 'FontSize',12)
    ylabel('g(x) [--:y=x]', 'FontSize',12)
    title('            x =g(x):The function and path to the root
(*:initial guess ; o:root)', 'FontSize',12)
end

if iter >=itermax
    disp('Warning:Maximum iterations reached.')
end
```

程序运行的输入和输出结果:

```
Calculating the friction factor from Colebrook equation
  Reynolds No.       =2e5
  Function Containing the Colebrook equation:'colebrookg'
  Starting value of b=0.0001
  Iteration          x            g(x)
       1           0.0001     0.00108666
       2       0.00108666     0.00080751
       3       0.00080751     0.00083597
       4       0.00083597    0.000832573
       5      0.000832573    0.000832971
  Friction factor, b=0.0008330
```

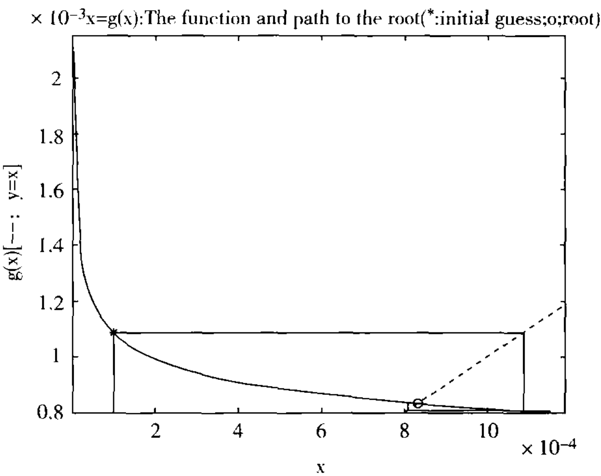


图5.11 逐次代换法求根的搜索路径

下面的第三个程序是用线性插值法求解同样的 Colebrook 非线性方程。

例5.2c 应用线性插值法求解非线性方程。

问题陈述：

解 Colebrook 方程，求尿液导管的摩擦系数。如下是 Colebrook 方程的$f(x)=0$的形式：

$$\frac{1}{\sqrt{b}}-4.07\log(Re\sqrt{b})+0.06=0$$

解题思路：

这里用式（5.10）的线性插值法，反复进行迭代计算，直到前后连续两次迭代得到的两个近似根之差小于某个默认值（如 10^{-6}）为止。并作图显示收敛过程，以便深入了解方程解的收敛性。

```
% example5_2c.m
% This program solves a nonlinear equation using the
% Linear Interpolation method.
% It calculates the friction factor from the Colebrook equation.

clc; clear all;

disp('Calculating the friction factor from Colebrook equation')
% Input
Re = input('\n Reynolds No.          = ');
fname = input('\n Function containing the Colebrook equation:');
%  Linear Interpolation method
b1 = input(' First  starting value b = ');
b2 = input(' Second starting value b = ');
b = LI(fname,b1,b2,[ ],2,Re);
fprintf('\n Friction factor, b = % 8.7f\n',b)
```

包含被求解方程的 MATLAB 函数
注：此处的 MATLAB 函数 colebrook.m 与例 5.2a 中的相同，故省略。

MATLAB 函数：实现线性插值法，并作图（见图 5.12）显示求根的搜索路径。该程序由 Constantinides 和 Mostoufi（1999 年）开发完成。

```
function x = LI(FunFcn,x1,x2,tol,trace,varargin)
% LI Finds a zero of a function by the linear interpolation method.
%
%   LI('F',X1,X2) finds a zero of the function described by the
%   M-file F.M. X1 and X2 are starting points where the function
%   has different signs at these points.
%   LI('F',X1,X2,TOL,TRACE) uses tolerance TOL for convergence
%   test. TRACE = 1 shows the calculation steps numerically and
%   TRACE = 2 shows the calculation steps both numerically and
```

```
%   graphically.
%
%   LI('F',X1,X2,TOL,TRACE,P1,P2,...) allows for additional
%   arguments which are passed to the function F(X,P1,P2,...).
%   Pass an empty matrix for TOL or TRACE to use the default
%   value.
%
%   See also FZERO, ROOTS, XGX, NR

% (c) N. Mostoufi & A. Constantinides
% January 1, 1999

% Initialization
if nargin < 4 | isempty(tol)
    tol = 1e-6;
end
if nargin < 5 | isempty(trace)
    trace = 0;
end
if tol == 0
    tol = 1e-6;
end
if(length(x1) > 1) | (~isfinite(x1)) | (length(x2) > 1) |...
        (~isfinite(x2))
    error('Second and third arguments must be finite scalars.')
end
if trace
    header = ' Iteration      x           f(x)';
    disp(header)
end
f1 = feval(FunFcn,x1,varargin{:});
f2 = feval(FunFcn,x2,varargin{:});

iter = 0;
if trace
    % Display initial values
```

```
    fprintf('% 5.0f  % 13.6g % 13.6g \n',iter, [x1 f1])
    fprintf('% 5.0f  % 13.6g % 13.6g \n',iter, [x2 f2])
    if trace==2
        xpath=[x1 x1 x2 x2];
        ypath=[0 f1 f2 0];
    end
end

if f1<0
    xm=x1;
    fm=f1;
    xp=x2;
    fp=f2;
else
    xm=x2;
    fm=f2;
    xp=x1;
    fp=f1;
end

iter=iter+1;
itermax=100;
x=xp;
x0=xm;

% Main iteration loop
while abs(x-x0)>tol & iter<=itermax
    x0=x;
    x=xp-fp * (xm-xp) / (fm-fp);
    fnk=feval(FunFcn,x,varargin{:});

    if fnk<0
        xm=x;
        fm=fnk;
    else
```

```
        xp = x;
        fp = fnk;
    end

    % Show the results of calculation
    if trace
        fprintf('% 5.0f  % 13.6g % 13.6g \n',iter, [x fnk])
        if trace ==2
            xpath =[xpath xm xm xp xp];
            ypath =[ypath 0 fm fp 0];
        end
    end
    iter = iter +1;
end

if trace ==2
    % Plot the function and path to the root
    xmin = min(xpath);
    xmax = max(xpath);
    dx = xmax - xmin;
    xi = xmin - dx/10;
    xf = xmax + dx/10;
    yc =[];
    for xc = xi:(xf - xi)/99:xf
        yc =[yc feval(FunFcn,xc,varargin{:})];
    end
    xc = linspace(xi,xf,100);
    ax = linspace(0,0,100);

    plot(xc,yc,xpath,ypath,xc,ax,xpath(2:3),ypath(2:3),' * ',x,
fnk,'o')
    axis([xi xf min(yc) max(yc)])
    xlabel('x')
    ylabel('f(x)')
    title('Linear Interpolation:The function and path to the root
```

```
(*:initial guess ; o:root)')
end

if iter >= itermax
    disp('Warning:Maximum iterations reached.')
end
```

程序运行的输入和输出结果

```
Calculating the friction factor from Colebrook equation

  Reynolds No.        =2e5

  Function containing the Colebrook equation:'colebrook'
  First  starting value b=0.0003
  Second starting value b=0.0015
  Iteration           x            f(x)
        0          0.0003       25.1637
        0          0.0015      -10.0267
        1      0.00115809      -5.93487
        2      0.00099433      -3.29705
        3     0.000913895      -1.75924
        4     0.000873781     -0.917111
        5     0.000853604     -0.472089
        6     0.000843409     -0.241398
        7     0.000838246     -0.123012
        8     0.000835628    -0.0625739
        9     0.000834299    -0.0318015
       10     0.000833625    -0.0161549

  Friction factor, b=0.0008336
  Friction factor, b=0.0008360
```

结果讨论

这里，我们用了 3 种不同的方法求解 Colebrook 非线性方程，例 5.2a 是 N-R 法，例 5.2b 是逐次代换法，例 5.2c 是线性插值法。其中，N-R 法是最常用的方法，但是，其搜索过程对于根的初始估计值的选定很敏感。本题中，逐次代换法

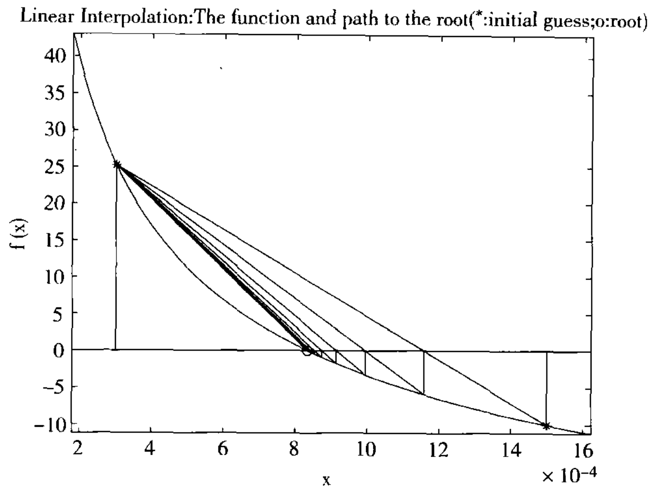


图 5.12　线性插值法求根的搜索路径

确定根的迭代步数最少，它也是 3 种方法中最简单的一种，不需要计算函数的导数。线性插值法需要给定两个初始估计值，用于设定搜索范围的起点和终点，并且，这种方法所需的迭代次数最多。3 种方法究竟选用哪种最好，这取决于非线性方程的性质以及初始估计值。一般，要求读者掌握 N-R 法。

例 5.3　应用 N-R 法求解 Michaelis-Menten 动力学方程（分子生物工程问题举例）。

问题陈述：

Michaelis-Menten 模型（参见 5.3.1 节）的解析解为

$$K_m \ln\left(\frac{s_0}{s}\right) + (s_0 - s) = V_{max} t$$

其中，K_m 等于 0.5mM，是酶促反应速率达到最大反应速率一半时的底物浓度；最大底物消耗率 V_{max} 等于 0.5mM/min；初始底物浓度 s_0 为 1.0mM。请用 N-R 法求 0～200*min* 范围内底物浓度随时间变化的曲线，时间增量设为 0.1min。

解：

MATLAB 程序如下：

```
% example5 _ 3. m
% This program calculates the roots of the substrate
% consumption equation governed by Michealis-Menten kinetics.

clc; clear all;
```

```
disp('Time     Concentration')
x0 =0.95;
% initialize the counter and increment by unity during each
% iteration. We could use a while loop below; a FOR loop is
% preferred because it initializes the counter and increments
% automatically at each iteration
for i =1:201
    t(i) =i -1;
    sub(i) =NR('enzymeconsump',x0,[],[],t,1.0,0.05,0.5);
    fprintf('% 3g         % g \n',t(i),sub(i))
    x0 =sub(i);
end
figure(1); plot(t,sub)
xlabel('Time, min'); ylabel('Concentration, mM')
title('Michaelis-Menten reaction')
```

包含 Michaelis-Menten 方程的 MATLAB 函数为

```
function y =enzymeconsump(s,t,so,vmax,Km)
%  enzymeconsump.m file
y =Km * log(so/s) + (so -s) -vmax * t;
```

程序运行的输出结果为（只显示图形输出）:

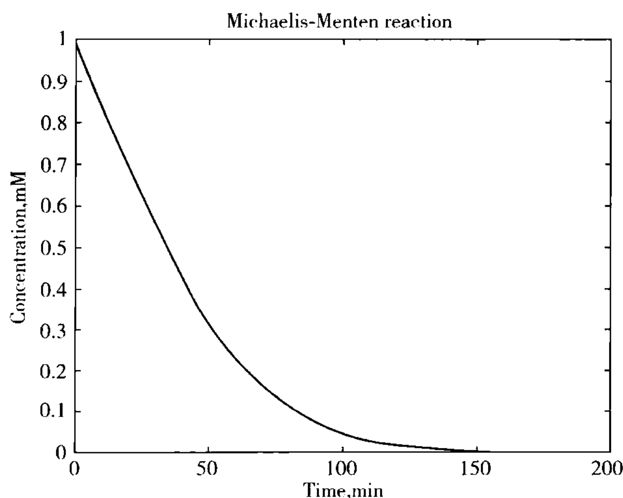


图 5.13 Michaelis-Menten 酶反应

5.7　牛顿法求解非线性方程组

如果模型中包含两个或更多个联立非线性方程，那么可以扩展 N-R 法，同时求解这些方程。假设存在两个未知变量 x_1 和 x_2，以及包含这两个变量的两个函数：

$$\begin{aligned} f_1(x_1, x_2) &= 0 \\ f_2(x_1, x_2) &= 0 \end{aligned} \tag{5.20}$$

式中　f_1，f_2——非线性函数。

利用二维泰勒级数，在初始估计值 $x_1^{(1)}$ 和 $x_2^{(1)}$ 处展开这两个函数，（这里的上标（1）表示搜索根的第一次迭代），有

$$f_1(x_1, x_2) = f_1(x_1^{(1)}, x_2^{(1)}) + \left.\frac{\partial f_1}{\partial x_1}\right|_{x(1)}(x_1 - x_1^{(1)}) + \left.\frac{\partial f_1}{\partial x_2}\right|_{x(1)}(x_2 - x_2^{(1)}) + \cdots \tag{5.21}$$

$$f_2(x_1, x_2) = f_2(x_1^{(1)}, x_2^{(1)}) + \left.\frac{\partial f_2}{\partial x_1}\right|_{x(1)}(x_1 - x_1^{(1)}) + \left.\frac{\partial f_2}{\partial x_2}\right|_{x(1)}(x_2 - x_2^{(1)}) + \cdots$$

前后两次迭代得到的 x 值之间的差定义为校正变量，即：

$$\begin{aligned} \delta_1^{(1)} &= x_1 - x_1^{(1)} \\ \delta_2^{(1)} &= x_2 - x_2^{(1)} \end{aligned} \tag{5.22}$$

使式（5.21）左边为 0，并截去泰勒级数中二阶以及二阶以上的高阶导数项，得到方程：

$$\begin{aligned} \left.\frac{\partial f_1}{\partial x_1}\right|_{x(1)}\delta_1^{(1)} + \left.\frac{\partial f_1}{\partial x_2}\right|_{x(1)}\delta_2^{(1)} &= -f_1(x_1^{(1)}, x_2^{(1)}) \\ \left.\frac{\partial f_2}{\partial x_1}\right|_{x(1)}\delta_1^{(1)} + \left.\frac{\partial f_2}{\partial x_2}\right|_{x(1)}\delta_2^{(1)} &= -f_2(x_1^{(1)}, x_2^{(1)}) \end{aligned} \tag{5.23}$$

这是一个线性代数方程组，未知数为 $\delta_1^{(1)}$ 和 $\delta_2^{(1)}$。求解 $\delta_1^{(1)}$ 和 $\delta_2^{(1)}$，结果如下：

$$\begin{aligned} \delta_1^{(1)} &= -\frac{\left[f_1\dfrac{\partial f_2}{\partial x_2} - f_2\dfrac{\partial f_1}{\partial x_2}\right]}{\left[\dfrac{\partial f_1}{\partial x_1}\dfrac{\partial f_2}{\partial x_2} - \dfrac{\partial f_1}{\partial x_2}\dfrac{\partial f_2}{\partial x_1}\right]} \\ \delta_2^{(1)} &= -\frac{\left[f_2\dfrac{\partial f_1}{\partial x_1} - f_1\dfrac{\partial f_2}{\partial x_1}\right]}{\left[\dfrac{\partial f_1}{\partial x_1}\dfrac{\partial f_2}{\partial x_2} - \dfrac{\partial f_1}{\partial x_2}\dfrac{\partial f_2}{\partial x_1}\right]} \end{aligned} \tag{5.24}$$

将前一个估计值与校正变量相加，就得到新的估计值，即

$$x_i^{(n+1)} = x_i^{(n)} + \delta_i^{(n)} \tag{5.25}$$

每次迭代之后，用此式更新估计值。反复执行这个迭代过程，直到所有的校正变量都小于某个规定的允许偏差为止。

如果非线性方程组含有 2 个以上的方程，那么可以用同样的方法得到一个线性代数方程组，用矩阵形式表示就是：

$$\begin{bmatrix} \dfrac{\partial f_1}{\partial x_1} & \cdots\cdots & \dfrac{\partial f_1}{\partial x_k} \\ & \cdots\cdots & \\ \dfrac{\partial f_k}{\partial x_1} & \cdots\cdots & \dfrac{\partial f_k}{\partial x_k} \end{bmatrix} \begin{bmatrix} \delta_1 \\ \cdot \\ \cdot \\ \cdot \\ \delta_k \end{bmatrix} = - \begin{bmatrix} f_1 \\ \cdot \\ \cdot \\ \cdot \\ f_k \end{bmatrix} \tag{5.26}$$

还可以利用矩阵矢量符号记为

$$\boldsymbol{J\delta} = -\boldsymbol{f} \tag{5.27}$$

式中　$\boldsymbol{J}$——偏微分的雅可比矩阵；

$\boldsymbol{\delta}$——校正矢量；

$-\boldsymbol{f}$——函数矢量。

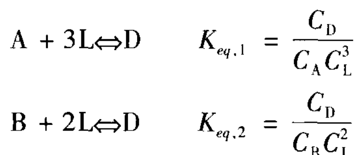
读者应该注意，这是线性代数方程组。为了求取 $\boldsymbol{\delta}$ 的值，首先要用数值微分的方法计算 $\boldsymbol{J}$ 矩阵，再求 $\boldsymbol{J}$ 矩阵的逆。只要 $\boldsymbol{J}$ 矩阵是非奇异矩阵，矩阵求逆可以用 MATLAB 指令 inv 完成。用 MATLAB 指令 det 求 $\boldsymbol{J}$ 矩阵行列式的值，如果行列式的值不等于 0，就可以判断该矩阵是非奇异矩阵。

例 5.4　应用牛顿法求解非线性方程组，计算受体与配体结合的动力学过程中受体的结合率。

问题陈述：

哺乳动物细胞的细胞膜受体可以与特定的配体（如激素、生长因子等）结合，从而触发细胞内的信息传导（Lauffenburger 和 Linderman，1993）。多聚受体具有多个不同的亚基或抗原表面，可以结合多个配体，因而使得受体与配体的结合反应呈现非线性。假设细胞膜有两种多聚受体，一种可以结合 3 个配体分子（即三聚体），另一种可以结合 2 个配体分子（即二聚体）。两者的可逆结合反应方程式建立如下：

设 C_A 为三聚体受体的浓度，C_B 为二聚体受体的浓度，C_L 为配体的浓度，C_D 为所有已与配体结合的受体总浓度。则



$$A + 3L \Leftrightarrow D \qquad K_{eq,1} = \frac{C_D}{C_A C_L^3}$$

$$B + 2L \Leftrightarrow D \qquad K_{eq,2} = \frac{C_D}{C_B C_L^2}$$

A、B 和 L 的初始浓度设为：$C_{A,0}=5000$/细胞，$C_{B,0}=10000$/细胞，$C_{L,0}=10^{-7}$ M。（1M = 1 摩尔/升 = 6.023×10^{23} 分子/升。1 个细胞的体积等于 $4\times10^{-9}\text{cm}^3$，因此，$C_{L,0}=24000$/细胞。）

解：

求解思路分析：

设 x_1 和 x_2 分别为 A 和 B 两种受体中已与配体结合的量占各自总量的比例，则

$$C_A = 5000(1-x_1)$$
$$C_B = 10000(1-x_2)$$
$$C_D = 5000x_1 + 10000x_2$$
$$C_L = 24000 - 5000x_1 - 10000x_2$$

代入上面的方程，可得：

$$K_{eq,1} = (5000x_1 + 10000x_2)/\{5000(1-x_1)(24000-5000x_1-10000x_2)^3\}$$
$$K_{eq,2} = (5000x_1 + 10000x_2)/\{10000(1-x_2)(24000-5000x_1-10000x_2)^2\}$$

数值求解：

```
% example5_4.m
% Commands to solve the problem by calling on Newton's method.

clc; clear all;

% Input two guesses for x1 and x2, respectively
x0 =[.5 .2];
% Solution is provided by calling Newton.m and the prescribed
% function, with starting estimate vector x0
[x, iter] =Newton('receptor_ligand_func',x0)
```

待求解的函数：

```
function f = receptor_ligand_func(x)  % This is saved as a -
Mfile.
x1 =x(1); x2 =x(2);
f(1) = (5000 * x1 +10000 * x2)/(5000 * (1 - x1) * (24000 - 5000 * x1 -
10000 * x2)^3) -7.19e -14;
f(2) = (5000 * x1 +10000 * x2)/(10000 * (1 - x2) * (24000 - 5000 * x1 -
```

```
10000 * x2)^2) - 6.0e - 10;
f = f';
```

应用牛顿法求解方程组的 MATLAB 函数

```
function [xnew , iter] = Newton(fnctn , x0 , rho , tol , varargin)
% NEWTON Solves a set of equations by Newton's method.
%
%   NEWTON('F',X0) finds a zero of the set of equations
%   described by the M-file F.M. X0 is a vector of starting
%   guesses.
%
%   NEWTON('F',X0,RHO,TOL) uses relaxation factor RHO and
%   tolerance TOL for convergence test.
%
%   NEWTON('F',X0,RHO,TOL,P1,P2,...) allows for additional
%   arguments which are passed to the function F(X,P1,P2,...).
%   Pass an empty matrix for RHO or TOL to use the default
%   value.

% (c) by N. Mostoufi & A. Constantinides
% January 1, 1999

% Initialization
if nargin < 4 | isempty(tol)
    tol = 1e - 6;
end
if nargin < 3 | isempty(rho)
    rho = 1;
end

x0 = (x0(:).')'; % Make sure it's a column vector
nx = length(x0);
x = x0 * 1.1;
xnew = x0;
iter = 0;
```

```
maxiter=100;
% Main iteration loop
while max(abs(x-xnew))>tol & iter<maxiter
    iter=iter+1;
    x=xnew;
    fnk=feval(fnctn,x,varargin{:});

    % Set dx for derivation
    for k=1:nx
        if x(k)~=0
            dx(k)=x(k)/100;
        else
            dx(k)=1/100;
        end
    end

    % Calculation of the Jacobian matrix
    a=x;
    b=x;
    for k=1:nx
        a(k)=a(k)-dx(k);  fa=feval(fnctn,a,varargin{:});
        b(k)=b(k)+dx(k);  fb=feval(fnctn,b,varargin{:});
        jacob(:,k)=(fb-fa)/(b(k)-a(k));
        a(k)=a(k)+dx(k);
        b(k)=b(k)-dx(k);
    end

    % Next approximation of the roots
    if det(jacob)==0
        xnew=x+max([abs(dx),1.1*tol]);
    else
        xnew=x-rho*inv(jacob)*fnk;
    end
end
```

```
if iter>=maxiter
    disp('Warning:Maximum iterations reached.')
end
```

程序运行的输出结果

```
x =
    0.3009
    0.0994
iter=6
```

5.8 本章学习要点

学完本章之后，读者应该掌握以下内容：

1）能够识别生物医学中的非线性代数方程问题，并且能够建立数值方法求解的解题公式。

2）本章的重点是应用 N-R 法求解非线性代数方程。同时也讲述了另外两种数值求解方法，即逐次代换法和线性插值法。

3）要充分认识 N-R 法求解过程中估计值正确收敛的重要性。正如例 5.2a 所示，初始估计值至关重要。如果初始估计值与方程根之间的距离很远，由于在这些距离根很远的点上，泰勒级数的高阶项很重要，截去高阶项的 N-R 公式给出的校正值就很不准确。另外，在搜索根的过程中如果遇到局部最大值或最小值，一阶导数趋于 0，则 N-R 法会发散，这是这种方法的一个缺点。读者要注意应用收敛准则判断欲求解问题的收敛性。

4）通过本章前 3 个例题的求解，读者要学会应用这些 MATLAB 程序求解非线性方程的方法。第 4 个例题说明了非线性方程组的求解方法。这些问题的求解都用到了 3 个程序：主程序、包含欲求解函数的子程序以及 N-R 法子程序（或者牛顿法子程序）。其中，主程序调用独立的函数子程序以及 N-R 法子程序或者牛顿法子程序。读者要练习编写类似的主程序用于求解具有不同复杂度的非线性代数方程。

5.9 习题

5.1 核磁共振成像（Magnetic Resonance Imaging，MRI）是一个迅速发展的生物医学成像领域。MRI 通过检测人体软组织中氢原子核的状态，建立人体组织的功能图像。常用的 MRI 线

圈是筒形电磁线圈。在信号采集过程中，利用电磁线圈产生的磁场将能量传递给被测物，同时检测随时间变化的磁通密度。假设线圈长为 l，半径为 a，线圈长轴在 y 轴方向，那么，线圈在任意 y 位置产生的磁通密度与线圈匝数 N、电流大小和真空磁导率 μ_0 有关，通常 $\mu_0 = 4\pi \times 10^{-7}$ H/m。则线圈的磁感应强度为

$$L = \frac{\mu_0 N^2 \pi a^2}{l^2}[(l^2 + a^2)^{1/2} - a]$$

假设 $N = 10$ 匝，$l = 20$cm，如果要求磁感应强度达到 2.6μH，请编写 MATLAB 程序应用 N-R 法计算线圈的半径 a 应该为多少。

5.2　当激光照射到生物组织上时，入射光波中会有部分被反射、吸收、散射和透射，光强随着入射距离的增加而减弱，扩散方程可以描述这种入射光的衰减过程。假设激光纤维发出的光源是均匀的，生物组织对入射光有吸收和散射作用，并且假设组织介质无限大。当散射为主时，在离光纤较远的 r 距离处光的能流率为（单位 W/cm^2）

$$\phi(r) = \frac{\phi_o}{4\pi D}\frac{e^{-r/\delta}}{r}$$

式中　δ——穿透深度，$\delta = \sqrt{D/\mu_a}$；

μ_a——组织吸收系数，$\mu_a = 0.1\text{cm}^{-1}$。

散射系数 $\mu_s = 100\text{cm}^{-1}$，组织的各向异性率 $g = 0.9$。请计算光强下降到源强度的 10% 的位置与激光点源的距离。（提示：$D =$ （1/3）$\mu_{t'}$；$\mu_{s'} = \mu_S$ $(1-g)$；$\mu_{t'} = \mu_a + \mu_{s'}$）

5.3　如下是描述生物组织中光传播的一个经验公式：

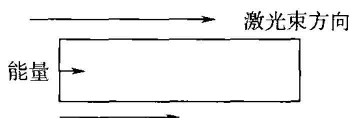
$$\frac{A-1}{A+1} = -1.440n_{rel}^{-2} + 0.710n_{rel}^{-1} + 0.688 + 0.0636n_{rel}$$

式中　A——组织的反射系数；

n_{rel}——组织与周围介质的折射率之比。

如果 n_{rel} 等于 1，则 A 接近于 1。如果要使反射系数 A 等于 4.0，请计算 n_{rel} 的值。

5.4　如图所示的一维激光照射消融可以用微分方程来描述，方程的解为



$$B\lambda = \frac{2}{\sqrt{\pi}}B\sqrt{\tau_{ab}} + e^{B^2\tau_{ab}}erfc[B\sqrt{\tau_{ab}}] - 1$$

式中　B——无量纲吸收参数；

λ——无量纲加热参数；

τ_{ab}——无量纲消融起始时间；

$erfc$——补余误差函数。

请计算（$B\lambda$）$= 28.359$ 时的（$B\sqrt{\tau_{ab}}$）值。

5.5　细胞在二维基质上迁移的位移平方的均值可以用如下 Dunn 方程计算（Dunn，1983）：

$$< d^2 > = 2S^2[Pt - P^2(1 - e^{-t/P})]$$

式中 $<d^2>$——细胞位移平方的均值；

S——细胞迁移速度的均方根；

P——细胞沿某方向运动的持续时间（单位 min）。

由化学诱导因素激发，白血球在多孔聚四氟乙烯人造血管材料上迁移的速度是 20μm/min（Chang 等人，2000）。如果要在 3h 内使一群白血球的平均位移平方达到 $4.3\times10^{-3}\text{cm}^2$，请计算所需的持续时间 P 为多少。

5.6 要使组织植入获得成功，其中的一个主要问题是要给植入组织供氧，也就是组织中要有充分接近的毛细血管，毛细血管中有携带氧分子的红血球流动。Fournier 在 1999 年提出了模拟这个问题的 Krogh 圆柱模型，他假设圆柱形的毛细血管周围围绕着组织细胞，在新鲜血流的驱动下氧和其他代谢物质沿着毛细血管轴向传输，同时穿过毛细血管壁径向扩散到组织中，并被组织中的细胞吸收。Krogh 圆柱问题的解产生了一个临界距离 r_{crit}，一旦超出这个临界距离，细胞就得不到扩散的氧等代谢物质。这个解的方程为

$$y^2\ln(y^2)-y^2+1-\left[\frac{4D_TC_o}{R_o(r_c+t_m)^2}\right]+4\frac{D_T}{Vr_c^2}[y^2-1]z+\frac{2D_T}{r_cK_o}[y^2-1]=0$$

其中，$y=\dfrac{r_{crit}}{r_c+t_m}$。

这里的各个参数为

$D_T=8\times10^{-6}\text{cm}^2/\text{s}$ 为组织代谢扩散率；

$V=0.005\text{cm/s}$ 为血浆流动速度；

$r_c=0.0005\text{cm}$ 为毛细血管半径；

$t_m=5\times10^{-5}\text{cm}$ 为毛细血管壁厚度；

$K_o=5.75\times10^{-5}\text{cm/s}$ 为总的代谢物质传输率；

$C_o=5\mu\text{mol/cm}^3$；

$R_o=0.01\mu\text{mol/(cm}^3\text{s)}$。

请应用 N-R 法解以上方程，求取临界距离 r_{crit} 随 z 变化的函数，z 的变化范围为 0.001 ~ 0.1cm，分辨率为 0.01cm。并作出 r_{crit} 随 z 变化的曲线。

5.7 酶是生物催化剂，可以固定在生化反应器的多孔基质中，与反应器内的底物扩散混合后发生反应。这种固定化酶反应器已经发展成为一种体外装置，用于选择性去除或者转化血液中的某些特殊成分，例如黄疸的胆红素和抗凝血的肝素等（Lavin 等人，1985；Sung 等人，1986）。由固定化酶颗粒上底物的质量守恒可以推导出反应效率 η 与 Thiele 模数 ϕ 之间的关系，ϕ 是底物反应率与底物扩散率之比，ϕ 值大表示扩散制定了总体底物消耗率，ϕ 值小则表示反应率制定了总体底物消耗率。对于一阶反应，反应效率 η 与 Thiele 模数 ϕ 之间的关系式为

$$\eta=\frac{1}{\phi}\left(\frac{1}{\tanh 3\phi}-\frac{1}{3\phi}\right)$$

请在代谢反应效率 30% ~60% 的变化范围内（递增步长 5%），计算 Thiele 模数 ϕ 值的变化并作图。

5.10 参考文献

Chang, C. C., Schloss, R. S., and Moghe, P. V. 2000. Quantitative Analysis of the Regulation of Leukocyte Chemosensory Migration by a Vascular Prosthetic Biomaterial. *J. Mater. Sci. Mater. Med.* **11**:337-344.

Constantinides, A. and Mostoufi, N. 1999. *Numerical Methods for Chemical Engineers with MATLAB Applications.* Upper Saddler River, NJ: Prentice Hall PTR.

Dunn, G. A. 1983. Characterising a kinesis response: time averaged measures of cell speed and directional persistence. *Agents and Actions* [Suppl.], **22**:14-33.

Enderle, J. D., Blanchard, S. M., and Bronzino, J. D. 2005. *Introduction to Biomedical Engineering.* 2nd Ed., San Diego, CA: Academic Press.

Fournier, R. L. 1999. *Basic Transport Phenomena in Biomedical Engineering.* Philadelphia, PA: Taylor & Francis.

Lauffenburger, D. A., and Linderman, J. J. 1993. *Receptors: Models for Binding, Trafficking, and Signaling.* New York: Oxford University Press.

Lavin, A., Sung, C., Klibanov, A.L., and Langer, R. 1985. A potential treatment for neonatal jaundice. *Science* **230**:543-545.

Sung, C., Lavin, A. , Klibanov, A. M., and Langer, R. 1986. An immobilized enzyme reactor for the detoxification of bilirubin. *Biotechnol. Bioeng.* **28**: 1531-1539.

第3部分　系统的动态行为

第6章　有限差分法、插值法和积分法

6.1　绪论

在科学研究和工程应用中最常见的数学模型是微分方程形式的模型，生物医学领域也是如此。具有单个独立变量的系统可以用常微分方程来模拟，有2个或2个以上独立变量的系统则要用偏微分方程。大多数微分方程，特别是非线性微分方程以及较大规模的微分方程组，没有解析解，需要用数值方法求解。

本书的第6章、第7章和第8章将讲述微分、积分、以及常微分方程和偏微分方程求解的数值方法。这些方法都基于有限差分的概念，因此，本章的前几节将介绍有限差分法，并推导有限差分和差分算子之间的关系，然后，应用这些概念推导差分公式和积分公式，并用数值方法求解常微分方程和偏微分方程。这里所介绍的内容基本上与Constantinides和Mostoufi（1999）著作中的内容相似。

有限差分法具有“双行道”的特性。如果某个微分方程已知，则应用有限差分法，通过计算有限个离散点上的函数值，可以求得微分方程的数值积分解；反过来，如果已知有限个数据，比如一组实验数据，也可以用有限差分法对这些数据进行微分或积分。但是，要注意，数值微分比数值积分的准确度要低。

有限差分法还有一个应用就是推导插值多项式，包括内插公式和外插公式。如果实验数据所对应的函数公式未知，可以用插值多项式表示实验数据。格雷戈里-牛顿（Gregory-Newton）插值公式（6.7节）等插值多项式是牛顿-科茨（Newton-Cotes）求积公式（6.10节）等数值积分的基础。

本章包括如下学习内容：

1）掌握有限差分的概念，并学会应用有限差分计算导数。

2）开发程序，计算函数或数据的数值积分和数值微分。

3）掌握插值多项式的概念，并用于推导积分公式。

6.2 符号算子

在微分计算中，导数的定义为

$$\left.\frac{\mathrm{d}f(x)}{\mathrm{d}x}\right|_{x_0} = f'(x_0) = \lim_{x \to x_0} \frac{f(x) - f(x_0)}{x - x_0} \tag{6.1}$$

在有限差分法中，$(x - x_0)$ 的值不趋于 0，而是一个有限的值，设该值为 h，即

$$h = x - x_0 \tag{6.2}$$

则导数的近似值为

$$f'(x_0) \cong \frac{f(x_0 + h) - f(x_0)}{h} \tag{6.3}$$

某些情况下，在区间 (a, b) 内存在点 ξ，该点的导数可以用式（6.3）准确计算。这就是微分中值定理。

微分中值定理：设 $f(x)$ 在区间 $a \leqslant x \leqslant b$ 中连续，并在区间 $a < x < b$ 内可微分，则至少存在一个 ξ 值 $a < \xi < b$，使得

$$f'(\xi) = \frac{f(b) - f(a)}{b - a} \tag{6.4}$$

该定理是微分计算的基础，也是有限差分法的基础。

在区间 $[x_0, x]$ 连续并可微的函数 $f(x)$ 可以用泰勒公式（参见第 3 章）表示为

$$f(x) = f(x_0) + (x - x_0)f'(x_0) + \frac{(x - x_0)^2 f''(x_0)}{2!} + \frac{(x - x_0)^3 f'''(x_0)}{3!} + \cdots + \frac{(x - x_0)^n f^{(n)}(x_0)}{n!} + R_n(x) \tag{6.5}$$

式中　$R_n(x)$ ——余项，此项表示无穷级数中第 $(n + 1)$ 项直到无穷项的总和。如果用第 n 项（包括 n 项）之前的有限项近似计算函数值，则余项就是截断误差。

利用微分中值定理可以证明，在区间 (x_0, x) 中存在一个点 ξ，使得余项为

$$R_n(x) = \frac{(x - x_0)^{n+1} f^{(n+1)}(\xi)}{(n + 1)!} \tag{6.6}$$

ξ 值是 x 的未知函数，所以，不可能求得余项（即截断误差项）的准确值。但是，这个余项提供了误差的上限，它是 $(x - x_0)^{n+1}$ 的函数，也是 $(n + 1)$ 阶导数的函数，是一个 $(n + 1)$ 阶的项，因此，我们在讨论截断误差时，将用余项的阶次来表示误差的大小，通常记为 $O(h^{n+1})$，此处的 h 由式（6.2）定义。

有限差分法的操作对象是离散的数据序列，这些离散数据可以是实验数据，如

$$y_{i-3} \quad y_{i-2} \quad y_{i-1} \quad y_i \quad y_{i+1} \quad y_{i+2} \quad y_{i+3}$$

也可以是连续函数 $y(x)$ 的离散值，如

$$y(x-3h) \quad y(x-2h) \quad y(x-h) \quad y(x) \quad y(x+h) \quad y(x+2h) \quad y(x+3h)$$

同样，可以是函数 $f(x)$ 的离散值，如

$$f(x-3h) \quad f(x-2h) \quad f(x-h) \quad f(x) \quad f(x+h) \quad f(x+2h) \quad f(x+3h)$$

所有这些例子中，因变量 y 或 f 的值都对应于等距分布的自变量 x 的值。图 6.1 用光滑函数 y （x） 为例显示了这个概念。

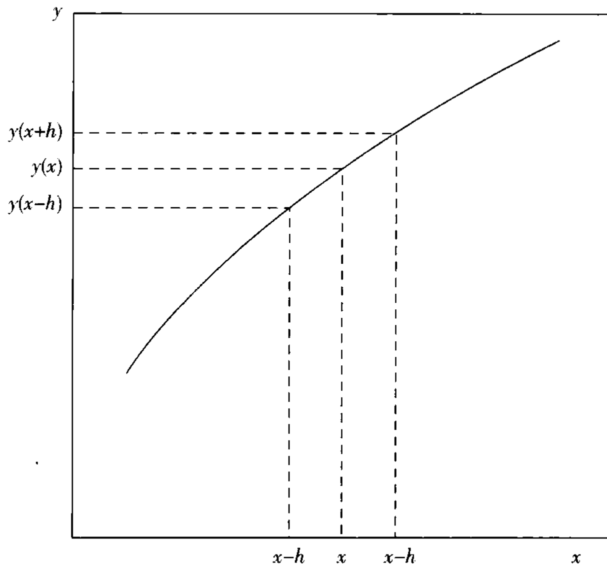


图 6.1　等距分布的自变量 x 对应的 $y(x)$ 函数的值

下面将利用这些离散序列定义一组用于微分计算和有限差分计算的线性符号算子，然后，再推导这些算子之间的关系。这些线性符号算子为

D = 微分算子；

∇ = 向后差分算子；

Δ = 向前差分算子；

δ = 中心差分算子

所有这些算子都满足代数运算的分配律、交换律和结合律，因此，可以作为代数变量来处理。

第一个是微分计算中熟知的微分算子 D，如果用于函数 $y(x)$，就是

$$Dy(x) = \frac{\mathrm{d}y(x)}{\mathrm{d}x} = y'(x) \tag{6.7}$$

记住这些概念，下面就可以推导向后差分算子、向前差分算子和中心差分算子。

6.3　向后有限差分

假设有一组沿 x 方向等距分布的 y 值：

$$y_{i-3} \quad y_{i-2} \quad y_{i-1} \quad y_i \quad y_{i+1} \quad y_{i+2} \quad y_{i+3}$$

也可以表示为

$$y(x-3h) \quad y(x-2h) \quad y(x-h) \quad y(x) \quad y(x+h) \quad y(x+2h) \quad y(x+3h)$$

在 i 处（即 x 处）y 的一阶向后差分定义为

$$\nabla y_i = y_i - y_{i-1}$$
$$\text{或者} \quad \nabla y(x) = y(x) - y(x-h) \tag{6.8}$$

在 i 处 y 的二阶向后差分定义为

$$\begin{aligned}\nabla^2 y_i &= \nabla(\nabla y_i) = \nabla(y_i - y_{i-1}) \\ &= \nabla y_i - \nabla y_{i-1} \\ &= (y_i - y_{i-1}) - (y_{i-1} - y_{i-2}) \\ &= y_i - 2y_{i-1} + y_{i-2}\end{aligned} \tag{6.9}$$

等价于 $y(x)$ 的：

$$\nabla^2 y(x) = y(x) - 2y(x-h) + y(x-2h) \tag{6.9a}$$

在 i 处 y 的三阶向后差分定义为

$$\begin{aligned}\nabla^3 y_i &= \nabla(\nabla^2 y_i) = \nabla(y_i - 2y_{i-1} + y_{i-2}) \\ &= \nabla y_i - 2\nabla y_{i-1} + \nabla y_{i-2} \\ &= (y_i - y_{i-1}) - 2(y_{i-1} - y_{i-2}) + (y_{i-2} - y_{i-3}) \\ &= y_i - 3y_{i-1} + 3y_{i-2} - y_{i-3}\end{aligned} \tag{6.10}$$

同样，可以得到更高的四阶和五阶向后差分：

$$\nabla^4 y_i = y_i - 4y_{i-1} + 6y_{i-2} - 4y_{i-3} + y_{i-4} \tag{6.11}$$

$$\nabla^5 y_i = y_i - 5y_{i-1} + 10y_{i-2} - 10y_{i-3} + 5y_{i-4} - y_{i-5} \tag{6.12}$$

这些有限差分中各项的系数对应于二项式 $(a-b)^n$ 的展开系数，n 就是有限差分的阶数。因此，n 阶向后有限差分的通式就可以表示为

$$\nabla^n y_i = \sum_{m=0}^{n} (-1)^m \frac{n!}{(n-m)!m!} y_{i-m} \tag{6.13}$$

注意，二项式展开系数之和总是等于 0。因此，可以用这条规律验证高阶差分公式是否正确。如图 6.2 所示，从杨辉三角的第 n 行可以得到 n 阶向后有限差分各项的系数，这样，就更容易记忆了。

杨辉三角	向后有限差分
1 1	$\nabla y_i = y_i - y_{i-1}$
1 2 1	$\nabla^2 y_i = y_i - 2y_{i-1} + y_{i-2}$
1 3 3 1	$\nabla^3 y_i = y_i - 3y_{i-1} + 3y_{i-2} - y_{i-3}$
1 4 6 4 1	$\nabla^4 y_i = y_i - 4y_{i-1} + 6y_{i-2} - 4y_{i-3} + y_{i-4}$
1 5 10 10 5 1	$\nabla^5 y_i = y_i - 5y_{i-1} + 10y_{i-2} - 10y_{i-3} + 5y_{i-4} - y_{i-5}$
1 6 15 20 15 6 1	$\nabla^6 y_i = y_i - 6y_{i-1} + 15y_{i-2} - 20y_{i-3} + 15y_{i-4} - 6y_{i-5} + y_{i-6}$

图 6.2 杨辉三角以及对应的向后有限差分

杨辉三角每行的第一个元素和最后一个元素都为 1，各个中间项则等于上一行两个相邻元素之和。另外，有限差分各项的符号是正负交替的。

例 6.1 用具有 h 阶误差的一阶向后有限差分表示一阶导数。

解:

用有限差分项表示导数的通用方法就是用第 3 章讲述的泰勒级数展开项来表示函数值，我们先用 y_i 处的泰勒公式表示 y_{i-1}:

$$y_{i-1} = y_i + (x_{i-1} - x_i)y_i' + \frac{(x_{i-1} - x_i)^2}{2}y_i'' + \cdots + \frac{(x_{i-1} - x_i)^n}{n!}y_i^n + R^{n+1}$$

截去二次项之后的项，得到:

$$y_{i-1} = y_i + (x_{i-1} - x_i)y_i' + \frac{(x_{i-1} - x_i)^2}{2}y_i'' + R^3$$

用 $(-h)$ 代替 $(x_{i-1} - x_i)$，并重排等式，得到:

$$y_i' = \frac{y_i - y_{i-1}}{h} + \frac{h}{2}y_i'' + R^3$$

由于误差以 y_i'' 项为主，把余项纳入，则有

$$\frac{dy_i}{dx} = y_i' = \frac{1}{h}(y_i - y_{i-1}) + O(h)$$

利用这个公式，就可以由向后有限差分计算 y 函数在 i 处的一阶导数。

$O(h)$ 项用于表示级数截去部分第一项的数量级。若 $h < 1.0$，并且函数光滑连续，则级数截去部分以其第一项为主。当 $h < 1.0$ 时，有

$$h > h^2 > h^3 > h^4 > \cdots > h^n$$

因此，具有高阶误差项 $O(h^n)$ 的泰勒公式的截断误差比较小，就可以更准确地计算导数的近似值。

若 $h > 1.0$，则

$$h < h^2 < h^3 < h^4 < \cdots < h^n$$

此时，具有高阶误差项的泰勒公式的截断误差就更大，计算的导数近似值就更不

准确。

由此可见，步长 h 的大小对于数值积分和数值微分的准确度和稳定性非常重要。本书第 7 章和第 8 章将详细讲述这个问题。

例 6.2　用具有 h^2 阶误差的向后有限差分表示一阶导数。

解：

先用 y_i 处的泰勒公式分别表示 y_{i-1} 和 y_{i-2}：

$$y_{i-1} = y_i + (x_{i-1} - x_i)y'_i + \frac{(x_{i-1} - x_i)^2}{2}y''_i + \cdots + \frac{(x_{i-1} - x_i)^n}{n!}y^n_i + R^{n+1}$$

$$y_{i-2} = y_i + (x_{i-2} - x_i)y'_i + \frac{(x_{i-2} - x_i)^2}{2}y''_i + \cdots + \frac{(x_{i-2} - x_i)^n}{n!}y^n_i + R^{n+1}$$

用 $(-h)$ 代替 $(x_{i-1} - x_i)$，用 $(-2h)$ 代替 $(x_{i-2} - x_i)$，并截去三次项之后的项，得到：

$$y_{i-1} = y_i - hy'_i + \frac{h^2}{2}y''_i - \frac{h^3}{6}y'''_i + R^4$$

$$y_{i-2} = y_i - 2hy'_i + \frac{(2h)^2}{2}y''_i - \frac{(2h)^3}{6}y'''_i + R^4$$

为了使误差项的阶数成为 $O(h^2)$，而不是 $O(h)$，下面消去这两个公式中的 y''_i 项，即将第一个公式乘以 4，再减第二个公式，得到：

$$4y_{i-1} - y_{i-2} = 3y_i - 2hy'_i + O(h^3)$$

重排等式之后，有：

$$\frac{\mathrm{d}y_i}{\mathrm{d}x} = y'_i = \frac{1}{2h}(3y_i - 4y_{i-1} + y_{i-2}) + O(h^2)$$

利用该公式，就可以由向后有限差分计算 y 函数在 i 处的一阶导数，并使其误差的阶数为 $O(h^2)$。

表 6.1 汇集了上述 2 个例题中推导的一阶导数公式，以及二阶、三阶和四阶求导公式。

可见，任何导数都可以根据要求的准确度用有限差分的形式表达。但是，所要求的准确度越高，需要计算的项数就越多。

如果已知等距 x 值上的一组 $y(x)$ 值，比如一组实验数据，就可以用这些公式计算函数 $y(x)$ 的微分。反之，这些公式也将在第 7 和第 8 章用于求解微分方程的数值积分。

表 6.1　用向后有限差分计算的导数

误差为 h 阶	
$\frac{\mathrm{d}y_i}{\mathrm{d}x} = \frac{1}{h}(y_i - y_{i-1}) + O(h)$	(6.14)

（续）

误差为 h 阶	
$\frac{d^2y_i}{dx^2} = \frac{1}{h^2}(y_i - 2y_{i-1} + y_{i-2}) + O(h)$	(6.15)
$\frac{d^3y_i}{dx^3} = \frac{1}{h^3}(y_i - 3y_{i-1} + 3y_{i-2} - y_{i-3}) + O(h)$	(6.16)
$\frac{d^4y_i}{dx^4} = \frac{1}{h^4}(y_i - 4y_{i-1} + 6y_{i-2} - 4y_{i-3} + y_{i-4}) + O(h)$	(6.17)
误差为 h^2 阶	
$\frac{dy_i}{dx} = \frac{1}{2h}(3y_i - 4y_{i-1} + y_{i-2}) + O(h^2)$	(6.18)
$\frac{d^2y_i}{dx^2} = \frac{1}{h^2}(2y_i - 5y_{i-1} + 4y_{i-2} - y_{i-3}) + O(h^2)$	(6.19)
$\frac{d^3y_i}{dx^3} = \frac{1}{2h^3}(5y_i - 18y_{i-1} + 24y_{i-2} - 14y_{i-3} + 3y_{i-4}) + O(h^2)$	(6.20)
$\frac{d^4y_i}{dx^4} = \frac{1}{h^4}(3y_i - 14y_{i-1} + 26y_{i-2} - 24y_{i-3} + 11y_{i-4} - 2y_{i-5}) + O(h^2)$	(6.21)

6.4 向前有限差分

向前有限差分的推导与向后有限差分的推导类似，所以，这里就直接给出向前差分公式的汇总。读者如果想了解完整的推导过程，可以参阅 Constantinides 和 Mostoufi 的著作（1999）。y 函数在 i 处的前几阶向前差分定义如下：

一阶： $$\Delta y_i = y_{i+1} - y_i \tag{6.22}$$

二阶： $$\Delta^2 y_i = y_{i+2} - 2y_{i+1} + y_i \tag{6.23}$$

三阶： $$\Delta^3 y_i = y_{i+3} - 3y_{i+2} + 3y_{i+1} - y_i \tag{6.24}$$

四阶： $$\Delta^4 y_i = y_{i+4} - 4y_{i+3} + 6y_{i+2} - 4y_{i+1} + y_i \tag{6.25}$$

五阶： $$\Delta^5 y_i = y_{i+5} - 5y_{i+4} + 10y_{i+3} - 10y_{i+2} + 5y_{i+1} - y_i \tag{6.26}$$

与向后有限差分相同，向前有限差分的系数也对应于二项式 $(a - b)^n$ 的展开系数，从杨辉三角的第 n 行可以方便地得到 n 阶向前差分各项的系数（见图 6.2）。

在 MATLAB 中，函数 diff（y）返回 y 的向前有限差分。n 阶向前有限差分的值可以用 diff（y，n）计算。例如：

```
>> y=[1 3 8 9 13 15]
```

```
y =
     1     3     8     9    13    15
>> diff(y)
ans =
     2     5     1     4     2
>> diff(y,2)
ans =
     3    -4     3    -2
>> diff(y,3)
ans =
    -7     7    -5
>> diff(y,4)
ans =
    14   -12
>> diff(y,5)
ans =
   -26
```

例 6.3　用具有 h 阶误差的向前有限差分表示一阶导数。

解：

先用 y_i 处的泰勒公式表示 y_{i+1}，并截去二次项之后的项，则：

$$y_{i+1} = y_i + (x_{i+1} - x_i)y'_i + \frac{(x_{i+1} - x_i)^2}{2}y''_i + R^3$$

用 h 代替 $(x_{i+1} - x_i)$，得

$$y_{i+1} = y_i + hy'_i + \frac{h^2}{2}y''_i + R^3$$

重排等式，得

$$y'_i = \frac{y_{i+1} - y_i}{h} + \frac{h}{2}y''_i + R^3$$

由于误差以 y''_i 项为主，即

$$\frac{\mathrm{d}y_i}{\mathrm{d}x} = \frac{1}{h}(y_{i+1} - y_i) + O(h)$$

利用这个公式，就可以由向前有限差分计算 y 函数在 i 处的一阶导数，其误差为 h 阶。

例 6.4　用具有 h 阶误差的向前有限差分表示二阶导数。

解：

二阶导数的有限差分公式中不能有一阶差分项，因此，需要建立两个方程，用来消去其中的一阶差分项。

与例 6.3 的方法相同，建立如下两个方程

$$y_{i+1} = y_i + (x_{i+1} - x_i)y'_i + \frac{(x_{i+1} - x_i)^2}{2}y''_i + R^3$$

$$y_{i+2} = y_i + (x_{i+2} - x_i)y'_i + \frac{(x_{i+2} - x_i)^2}{2}y''_i + R^3$$

用 h 代替 $(x_{i+1} - x_i)$，用 $2h$ 代替 $(x_{i+2} - x_i)$，得到

$$y_{i+1} = y_i + hy'_i + \frac{h^2}{2}y''_i + R^3$$

$$y_{i+2} = y_i + 2hy'_i + \frac{4h^2}{2}y''_i + R^3$$

用第二个方程减第一个方程再乘 2，得

$$y_{i+2} - 2y_{i+1} = -y_i + h^2y''_i + R^3$$

重排等式之后，有

$$\frac{d^2y_i}{dx^2} = y''_i = \frac{1}{h^2}(y_{i+2} - 2y_{i+1} + y_i) + O(h)$$

利用这个公式，就可以由向前有限差分计算 y 在 i 处的二阶导数，其误差为 h 阶。

表 6.2 汇集了上述 2 个例题中推导的一阶导数和二阶导数公式，以及三阶和四阶求导公式，还包括了相应的误差为 h^2 阶的一组方程。

应该指出，本节和上一节得到的所有有限差分求导公式中的系数之和都等于 0，这是所有这类有限差分都遵循的一条规则。

将表 6.1 与表 6.2 进行比较，可以看出，导数既可以用向后有限差分表示，也可以用向前有限差分表示，这两种公式的项数以及截断误差的阶数都很相似。具体采用哪一种有限差分取决于函数的波形及其边界条件，这将在第 7 章和第 8 章进一步讨论。

表 6.2　用向前有限差分计算的导数

误差为 h 阶	
$\frac{dy_i}{dx} = \frac{1}{h}(y_{i+1} - y_i) + O(h)$	(6.27)
$\frac{d^2y_i}{dx^2} = \frac{1}{h^2}(y_{i+2} - 2y_{i+1} + y_i) + O(h)$	(6.28)
$\frac{d^3y_i}{dx^3} = \frac{1}{h^3}(y_{i+3} - 3y_{i+2} + 3y_{i+1} - y_i) + O(h)$	(6.29)
$\frac{d^4y_i}{dx^4} = \frac{1}{h^4}(y_{i+4} - 4y_{i+3} + 6y_{i+2} - 4y_{i+1} + y_i) + O(h)$	(6.30)

（续）

误差为 h^2 阶	
$\frac{dy_i}{dx}=\frac{1}{2h}(-y_{i+2}+4y_{i+1}-3y_i)+O(h^2)$	(6.31)
$\frac{d^2y_i}{dx^2}=\frac{1}{h^2}(-y_{i+3}+4y_{i+2}-5y_{i+1}+2y_i)+O(h^2)$	(6.32)
$\frac{d^3y_i}{dx^3}=\frac{1}{2h^3}(-3y_{i+4}+14y_{i+3}-24y_{i+2}+18y_{i+1}-5y_i)+O(h^2)$	(6.33)
$\frac{d^4y_i}{dx^4}=\frac{1}{h^4}(-2y_{i+5}+11y_{i+4}-24y_{i+3}+26y_{i+2}-14y_{i+1}+3y_i)+O(h^2)$	(6.34)

6.5　中心有限差分

顾名思义，中心有限差分就是以主节点为中心，利用主节点左右两边距离为 $h/2$ 处的函数值计算差分。

在前两节用过的数据序列中增加节点间距中点的值，使其变成：

$$y_{i-2}\quad y_{i-1½}\quad y_{i-1}\quad y_{i-½}\quad y_i\quad y_{i+½}\quad y_{i+1}\quad y_{i+1½}\quad y_{i+2}$$

则 y 在 i 处的中心差分公式如下：

一阶：$\delta y_i = y_{i+½} - y_{i-½}$　　(6.35)

二阶：$\delta^2 y_i = y_{i+1} - 2y_i + y_{i-1}$　　(6.36)

三阶：$\delta^3 y_i = y_{i+1½} - 3y_{i+½} + 3y_{i-½} - y_{i-1½}$　　(6.37)

四阶：$\delta^4 y_i = y_{i+2} - 4y_{i+1} + 6y_i - 4y_{i-1} + y_{i-2}$　　(6.38)

五阶：$\delta^5 y_i = y_{i+2½} - 5y_{i+1½} + 10y_{i+½} - 10y_{i-½} + 5y_{i-1½} - y_{i-2½}$　　(6.39)

与前两种有限差分相同，中心有限差分的系数也对应于二项式 $(a-b)^n$ 的展开系数，也可以从杨辉三角的第 n 行得到 n 阶中心有限差分各项的系数（见图 6.2）。

应该注意，奇数阶中心差分用到了间距中点的函数值，而偶数阶中心差分只用到了节点上的值。如果奇数阶和偶数阶中心差分都要用，所需的 y 函数值的个数就要比向前差分和向后差分多 1 倍。这样就很不经济，尤其当这些数据必须通过实验测定时，增加的费用会更多。这个问题可以用平均算子的方法解决（参见 Constantinides 和 Mostoufi，1999）。

例 6.5　用具有 h^2 阶误差的中心有限差分表示一阶导数。

解：

先用泰勒公式表示 y_{i+1} 和 y_{i-1}：

$$y_{i+1}=y_i+(x_{i+1}-x_i)y'+\frac{(x_{i+1}-x_i)^2}{2}y_i''+\frac{(x_{i+1}-x_i)^3}{6}y'''+\cdots+\frac{(x_{i+1}-x_i)^n}{n!}y_i^n+R^{n+1}$$

$$y_{i-1}=y_i+(x_{i-1}-x_i)y'_i+\frac{(x_{i-1}-x_i)^2}{2}y_i''+\frac{(x_{i-1}-x_i)^3}{6}y_i'''+\cdots+\frac{(x_{i-1}-x_i)^n}{n!}y_i^n+R^{n+1}$$

用第一个公式减第二个公式，并且代入 $h=(x_{i+1}-x_i)$，$h=-(x_{i-1}-x_i)$，再进行截断，得到：

$$y_{i+1}-y_{i-1}=2hy'_i+\frac{h^3}{3}y_i'''+R^5$$

公式中的偶次方项被消去了，只剩下奇次方项。$\left(\frac{h^3}{3}y_i'''\right)$项比余项大得多，因此，可以求得 y' 为

$$\frac{\mathrm{d}y_i}{\mathrm{d}x}=y'_i=\frac{1}{2h}(y_{i+1}-y_{i-1})+O(h^2)$$

此公式用中心有限差分计算了 y 函数在 i 处的一阶导数。将这个公式与式（6.14）和式（6.27）进行比较，可见，在保留项项数相同的情况下，中心有限差分公式提高了计算的准确度。

例 6.6　用具有 h^2 阶误差的中心有限差分表示二阶导数。

解：

先用泰勒公式表示 y_{i+1} 和 y_{i-1}：

$$y_{i+1}=y_i+(x_{i+1}-x_i)y'+\frac{(x_{i+1}-x_i)^2}{2}y_i''+\frac{(x_{i+1}-x_i)^3}{6}y'''+\cdots+\frac{(x_{i+1}-x_i)^n}{n!}y_i^n+R^{n+1}$$

$$y_{i-1}=y_i+(x_{i-1}-x_i)y'_i+\frac{(x_{i-1}-x_i)^2}{2}y_i''+\frac{(x_{i-1}-x_i)^3}{6}y_i'''+\cdots+\frac{(x_{i-1}-x_i)^n}{n!}y_i^n+R^{n+1}$$

把两个公式相加，并且代入 $h=(x_{i+1}-x_i)$，$h=-(x_{i-1}-x_i)$，再进行截断，得到：

$$y_{i+1}+y_{i-1}=2y_i+h^2y''_i+\frac{h^4}{12}y_i^{iv}+R^6$$

公式中的奇次方项被消去了，只剩下偶次方项。$\left(\frac{h^4}{12}y_i^{iv}\right)$项比余项大得多，因此，可以求得 y' 为

$$\frac{\mathrm{d}^2y_i}{\mathrm{d}x^2}=y_i''=\frac{1}{h^2}(y_{i+1}-2y_i+y_{i-1})+O(h^2)$$

该公式用中心有限差分计算了 y 函数在 i 处的二阶导数，其误差为 h^2 阶。将这个公式与式（6.15）和式（6.28）比较，可见，对于同样数目的保留项，中心差分公式提高了计算的准确度。

表 6.3 汇集了例 6.5 和例 6.6 推导的一阶和二阶导数公式，以及三阶和四阶导数公式。

表6.3　用中心有限差分计算的导数

误差为 h^2 阶

$$\frac{dy_i}{dx}=\frac{1}{2h}(y_{i+1}-y_{i-1})+O(h^2) \tag{6.40}$$

$$\frac{d^2y_i}{dx^2}=\frac{1}{h^2}(y_{i+1}-2y_i+y_{i-1})+O(h^2) \tag{6.41}$$

$$\frac{d^3y_i}{dx^3}=\frac{1}{2h^3}(y_{i+2}-2y_{i+1}+2y_{i-1}-y_{i-2})+O(h^2) \tag{6.42}$$

$$\frac{d^4y_i}{dx^4}=\frac{1}{h^4}(y_{i+2}-4y_{i+1}+6y_i-4y_{i-1}+y_{i-2})+O(h^2) \tag{6.43}$$

误差为 h^4 阶

$$\frac{dy_i}{dx}=\frac{1}{12h}(-y_{i+2}+8y_{i+1}-8y_{i-1}+y_{i-2})+O(h^4) \tag{6.44}$$

$$\frac{d^2y_i}{dx^2}=\frac{1}{12h^2}(-y_{i+2}+16y_{i+1}-30y_i+16y_{i-1}-y_{i-2})+O(h^4) \tag{6.45}$$

$$\frac{d^3y_i}{dx^3}=\frac{1}{8h^3}(-y_{i+3}+8y_{i+2}-13y_{i+1}+13y_{i-1}-8y_{i-2}+y_{i-3})+O(h^4) \tag{6.46}$$

$$\frac{d^4y_i}{dx^4}=\frac{1}{6h^4}(-y_{i+3}+12y_{i+2}-39y_{i+1}+56y_i-39y_{i-1}+12y_{i-2}-y_{i-3})+O(h^4) \tag{6.47}$$

6.6　插值多项式

科学家和工程人员经常需要应用数值积分和数值微分方法解释和分析离散形式的实验数据，内插公式和外插公式就可以满足这些需求。当实验数据所对应的函数关系不太明确时，利用有限差分法导出的插值多项式可以描述实验数据。更重要的是，还可以利用这些多项式求取难以积分或难以微分的函数，比较容易地实现数值积分和数值微分。

假设在自变量 x 的 $(n+1)$ 个节点上函数 $f(x)$ 的值已知，为

$$\begin{array}{cc}
x_0 & f(x_0) \\
x_1 & f(x_1) \\
x_2 & f(x_2) \\
x_3 & f(x_3) \\
\cdot & \cdot \\
\cdot & \cdot \\
\cdot & \cdot \\
x_n & f(x_n)
\end{array}$$

如图 6.3 所示，这些值称为函数的基节点。

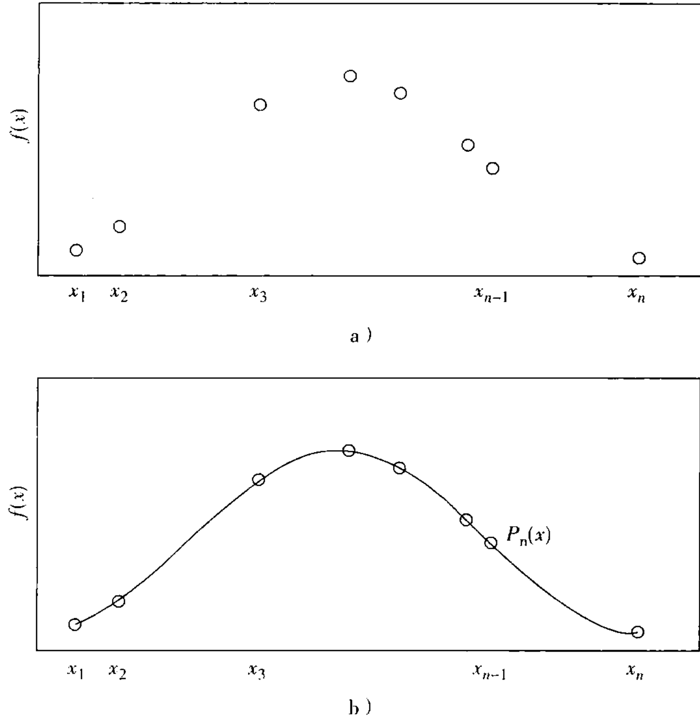


图 6.3

a）函数 $f(x)$ 的非等距基节点　b）非等距基节点及其插值多项式

求插值多项式的目标就是要确定如下形式的多项式

$$P_n(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + \cdots + a_nx^n \tag{6.48}$$

使得这个方程可以准确地拟合函数的基节点，并把这些基节点连接起来，形成光滑的曲线（见图 6.3b）。然后，就可以利用这个多项式求取基节点之间任意 x 值上的函数近似值。

在给定的（$n+1$）个已知基节点上，多项式必须满足等式：

$$P_n(x_i) = f(x_i) \qquad i = 0,1,2,\cdots,n \tag{6.49}$$

将已知的（$x_i, f(x_i)$）值代入式（6.48），可得（$n+1$）个联立线性代数方程，方程组的未知数就是多项式的系数 a_0，…，a_n。该线性代数方程组的解可以用第 4 章所介绍的方法求得，第 9 章会详细讨论这种解。但是，这个解会导致病态线性系统，所以，要用其他方法求插值多项式。

MATLAB 有几种求插值多项式的函数。函数 interp1 的一种调用形式是 y_i = interp1（x,y,x_i），输入自变量 x 和因变量 y 的各个值（即基节点的值），进行一维插值计算，求 x_i 对应的 y_i 值。该函数默认的插值方法是线性插值。不过，用

户也可以通过设置第 4 个输入参数来指定插值方法，例如：'nearest'表示最近邻插值，'linear'表示线性插值，'spline'表示三次样条插值，'cubic'表示三次多项式插值等。如果自变量不是等距分布，可以用函数 interp1q，由于不检查输入参数，该函数比 interp1 运算速度快。MATLAB 还有 spline 函数，它利用非扭结（not-a-knot）方法实现一维三次样条插值。根据需要，该函数还可以返回分段多项式的系数。函数 interp2、interp3 和 interpn 分别实现二维、三维和 n 维插值。

6.7　等距节点插值

本节将推导基于向前差分和向后差分的格雷戈里-牛顿公式（Gregory-Newton formula），用于已知数据或实验采样数据的插值计算。

6.7.1　格雷戈里-牛顿插值法

首先，假设等距 x 值上的一组已知的 $f(x)$ 函数值为

$x-3h$	$f(x-3h)$
$x-2h$	$f(x-2h)$
$x-h$	$f(x-h)$
x	$f(x)$
$x+h$	$f(x+h)$
$x+2h$	$f(x+2h)$
$x+3h$	$f(x+3h)$

图 6.4 显示了这些节点，同时表 6.4 列出了这些节点及其相应的一阶、二阶和三阶向后差分，表 6.5 则列出了这些节点及其相应的向前差分。

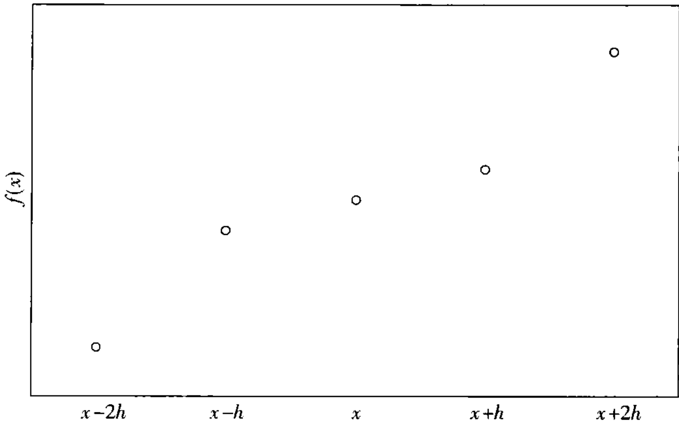


图 6.4　用于插值多项式的等距基节点

表 6.4　向后差分表

i	x_i	$f(x_i)$	$\nabla f(x_i)$	$\nabla^2 f(x_i)$	$\nabla^3 f(x_i)$
-3	$x-3h$	$f(x-3h)$			
-2	$x-2h$	$f(x-2h)$	$f(x-2h)-f(x-3h)$		
-1	$x-h$	$f(x-h)$	$f(x-h)-f(x-2h)$	$f(x-h)-2f(x-2h)+f(x-3h)$	
0	x	$f(x)$	$f(x)-f(x-h)$	$f(x)-2f(x-h)+f(x-2h)$	$f(x)-3f(x-h)+3f(x-2h)-f(x-3h)$

表 6.5　向前差分表

i	x_i	$f(x_i)$	$\Delta f(x_i)$	$\Delta^2 f(x_i)$	$\Delta^3 f(x_i)$
0	x	$f(x)$	$f(x+h)-f(x)$	$f(x+2h)-2f(x+h)+f(x)$	$f(x+3h)-3f(x+2h)+3f(x+h)-f(x)$
1	$x+h$	$f(x+h)$	$f(x+2h)-f(x+h)$	$f(x+3h)-2f(x+2h)+f(x+h)$	
2	$x+2h$	$f(x+2h)$	$f(x+3h)-f(x+2h)$		
3	$x+3h$	$f(x+3h)$			

利用 6.2 节定义的符号算子以及 6.4 节给出的向前有限差分公式，可以导出格雷戈里-牛顿向前插值公式，下面介绍推导过程。由符号算子的定义可知：

$$\Delta f(x) = f(x+h) - f(x) \tag{6.50}$$

因此

$$f(x+h) = (1+\Delta)f(x) \tag{6.51}$$

该公式使用 n 次之后，可得

$$f(x+nh) = (1+\Delta)^n f(x) \tag{6.52}$$

$(1+\Delta)^n$ 这一项可以用二项式级数展开为

$$(1+\Delta)^n = 1 + n\Delta + \frac{n(n-1)}{2!}\Delta^2 + \frac{n(n-1)(n-2)}{3!}\Delta^3 + \frac{n(n-1)(n-2)(n-3)}{4!}\Delta^4 + \cdots \tag{6.53}$$

于是，式（6.52）变为

$$f(x+nh) = f(x) + n\Delta f(x) + \frac{n(n-1)}{2!}\Delta^2 f(x) + \frac{n(n-1)(n-2)}{3!}\Delta^3 f(x) + \frac{n(n-1)(n-2)(n-3)}{4!}\Delta^4 f(x) + \cdots \tag{6.54}$$

其中，n 为正整数，二项式级数有 $(n+1)$ 项，因此，上式是一个 n 次多项式。如果已知函数 f 的 $(n+1)$ 个基节点的值，则该多项式应该与所有 $(n+1)$ 个基节点完全拟合。假设 $(n+1)$ 个基节点为 $(x_0, f(x_0))$，$(x_1, f(x_1))$，…，$(x_n, f(x_n))$，其中 $(x_0, f(x_0))$ 为主节点，x_i 定义为

$$x_i = x_0 + ih \tag{6.55}$$

于是，其他点与主节点之间的距离就可以表示为 $(x-x_0)$，此时 n 不再是整数，其值为

$$n = \frac{x-x_0}{h} \tag{6.56}$$

写出式（6.54）在原点（即 $x = x_0$）处的方程，再将式（6.56）代入，可得

$$f(x) = f(x_0) + \frac{(x-x_0)}{h}\Delta f(x_0) + \frac{(x-x_0)(x-x_1)}{2!h^2}\Delta^2 f(x_0) + \frac{(x-x_0)(x-x_1)(x-x_2)}{3!h^3}\Delta^3 f(x_0) + \frac{(x-x_0)(x-x_1)(x-x_2)(x-x_3)}{4!h^4}\Delta^4 f(x_0) + \cdots \tag{6.57}$$

这就是格雷戈里-牛顿向前插值公式。其通式为

$$f(x) = f(x_0) + \sum_{k=1}^{n}\left(\prod_{m=0}^{k-1}(x - x_m)\right)\frac{\Delta^k f(x_0)}{k!h^k} \tag{6.58}$$

利用向后差分，通过同样的推导过程，可得格雷戈里-牛顿向后插值公式：

$$f(x) = f(x_0) + \frac{(x - x_0)}{h}\nabla f(x_0) + \frac{(x - x_0)(x - x_{-1})}{2!h^2}\nabla^2 f(x_0) + \frac{(x - x_0)(x - x_{-1})(x - x_{-2})}{3!h^3}\nabla^3 f(x_0) + \frac{(x - x_0)(x - x_{-1})(x - x_{-2})(x - x_{-3})}{4!h^4}\nabla^4 f(x_0) + \cdots \tag{6.59}$$

其通式为

$$f(x) = f(x_0) + \sum_{k=1}^{n}\left(\prod_{m=0}^{k-1}(x - x_{-m})\right)\frac{\nabla^k f(x_0)}{k!h^k} \tag{6.60}$$

当 n 为正整数时，式（6.53）的二项式级数只有（$n+1$）项，是有限级数。而格雷戈里-牛顿插值公式中的 n 通常不是整数，因此，这些插值多项式有无限项。由代数运算可知，如果 $|\Delta| \leqslant 1$，那么，当项数越来越大时，$(1+\Delta)^n$ 的二项式级数收敛于 $(1+\Delta)^n$ 的值。也就是说有限差分必须很小。平滑函数或者基节点之间靠得很近（即 h 很小）的函数满足这个条件。当然，每个公式能够使用的项数取决于现有的已知数据中可以求得的有限差分的最高阶数。通常，对于等距数据，紧密分布的大量数据点的插值准确度最高。

对于给定的一组数据点，尽可能选择主节点靠近需要求值的点，使 $(x - x_0) < h$，可以进一步提高插值的准确度。如果满足 $(x - x_0) < h$ 这个条件，则插值公式就可以使用尽可能多的项数，也就是，公式中有限差分的数目最大。公式误差的阶数等于序列中第一个截去项所含有的有限差分的阶数。由表 6.4 可知，表中下部分的节点具有最大数目的向后差分，而表 6.5 则显示表中上部分的节点具有最大数目的向前差分。因此，向前差分公式应该用于靠近表上部分的节点之间的插值，而向后差分公式则应该用于靠近表下部分的节点之间的插值。

例 6.7 应用格雷戈里-牛顿法实现等距数据的插值

问题陈述：

某住院病人从晚上到第二天早晨体温上升很大，原因未知，护士发现了这个情况，并用了药。为了分析这个体温变化过程，需要确定病人体温达到最高点的时间以及最高体温值。体温由计算机每小时记录一次，表 6.6 列出了记录时间和相应的体温值。请编写一个通用的 MATLAB 函数，利用格雷戈里-牛顿向前插值

公式求 n 阶一维插值，并用于求解本题。

解：

该 MATLAB 函数要应用格雷戈里-牛顿向前插值的通用公式（6.58）实现 n 阶插值，因此，函数至少要有（$n+1$）个基节点输入参数。

程序说明：

本题使用 Constantinides 和 Mostoufi（1999）开发的格雷戈里-牛顿向前插值 MATLAB 函数 gregory_newton.m。该函数调用的一般形式为 y_i = gregory_newton(x, y, x_i, n)，其中，第一、二个输入参数为基节点坐标值，第三个输入参数是需要插值计算的因变量所对应的自变量矢量，第四个输入参数 n 是插值的阶数。如果不输入第四个参数，函数就使用线性插值法。如果要进行二阶及二阶以上的高阶插值，就需要输入第四个参数 n 的值。

开始时，函数首先检查输入参数。x 和 y 两个基节点坐标矢量的维数要相等。自变量矢量必须是单调增减的，如果不是，函数就中止运行。插值阶数不能大于基节点之间间隔的数目（即基节点个数减1)，如果插值阶数超过最大值，则函数显示警告信息，并按照最大插值阶数继续运行程序。完成这些检查之后，函数按照式（6.58）执行插值计算。

主程序 example6_7.m 是用于求解本题的。程序先要求用户输入时间矢量（即自变量）、病人体温矢量（即因变量）、以及插值阶数。然后，调用函数 gregory_newton.m 计算出记录体温之间的插值体温，并找出最大值。读者可以用不同的插值阶数重复运行该程序。

表 6.6　病人体温

时间（a.m.）	体温/°F	时间（a.m.）	体温/°F
1	98.9	7	104.0
2	99.5	8	104.1
3	99.9	9	102.5
4	101.3	10	101.2
5	101.6	11	100.5
6	102.5	12	100.2

程序

```
% example6_7.m-Interpolation of the time-temperature data
% given in Table 6.6 by Gregory-Newton forward interpolation
% formula to find the maximum temperature and the time this
```

```
% maximum occurred

clc; clear all;

% Input data
time = input(' Vector of time = ');
temp = input(' Vector of temperature = ');
%  Vector of time for interpolation
ti = linspace(min(time),max(time));
redo = 1;
while redo
    disp(' '); n = input(' Order of interpolation = ');
    te = gregory_newton(time,temp,ti,n);  % Interpolation
    [max_temp,k] = max(te);
    max_time = ti(k);
% Show the results
    fprintf('\n  Maximum  temperature  of  %4.1f  F  reached  at
%4.2f.\n',max_temp,max_time)
% Show the results graphically
    figure(1); plot(time,temp,'o',ti,te)
    title('Patient''s Temperature Profile')
    xlabel('Time (a.m.)'); ylabel('Temperature (deg F)')
    axis([1 12 98 105])
    disp(' ')
    redo = input(' Repeat the calculation (1/0):');
end
```

完成格雷戈里-牛顿插值的 MATLAB 函数

```
function yi = gregory_newton(x,y,xi,n)
%  gregory_newton One dimensional interpolation.
%
%   YI = Gregory_newton(X,Y,XI,N) applies the Nth-order
%   Gregory-Newton forward interpolation to find YI, the
%   values of the underlying function Y at the points in
%   the vector XI. The vector X specifies the points at
%   which the data Y is given
```

```
%
%   YI = Gregory_newton(X,Y,XI) is equivalent to the
%   linear interpolation
%
%   See also INTERP1, NATURALSPLINE, Lagrange, SPLINE, INTERP1Q

% (c) N. Mostoufi & A. Constantinides
% January 1, 1999
% Initialization
if nargin < 3
    error('Invalid number of inputs.')
end

% Check x for equal spacing and determining h
if min(diff(x)) ~= max(diff(x))
    error('Independent variable is not monotonic.')
else
    h = x(2) - x(1);
end

x = (x(:).')';     % Make sure it's a column vector
y = (y(:).')';     % Make sure it's a column vector
nx = length(x);
ny = length(y);
if nx ~= ny
    error('X and Y vectors are not the same size.');
end

% Check the order of interpolation
if nargin == 3 |n < 1
    n = 1;
end
n = floor(n);
if n >= nx
    fprintf('\nNot enough data points for % 2d-order interpolation.',n)
    fprintf('\n% 2d-order interpolation will be performed instead. \
```

```
n',nx -1)
    n =nx -1;
end

deltax(1,1:length(xi)) =ones(1,length(xi));
% Locating the required number of base points
for m =1:length(xi)
    dx =xi(m) -x;
    % Locating xi
    [dxm , loc(m)] =min(abs(dx));
    % locating the first base point
    if dx(loc(m)) <0
        loc(m) =loc(m) -1;
    end
    if loc(m) <1
        loc(m) =1;
    end
if loc(m) +n >nx
    loc(m) =nx -n;
    end
    deltax(2:n +1,m) =dx(loc(m):loc(m) +n -1);
    ytemp(1:n +1,m) =y(loc(m):loc(m) +n);
end

% Interpolation
yi =y(loc)';
for k =1:n
    yi =yi +prod(deltax(1:k +1,:)) .* diff(ytemp(1:k +1,:),k) /
...
        (gamma(k +1) * h^k);
end
```

程序运行的输入以及输出结果

```
>>example6_7
 Vector of time =[1:12]
```

```
    Vector of temperature =[98.9, 99.5, 99.9, 101.3, 101.6, 102.5,
104.0, 104.1, 102.5, 101.2, 100.5, 100.2]

    Order of interpolation =2

  Maximum temperature of 104.3 F reached at 7.56

  Repeat the calculation (1/0):0
```

结果讨论

图 6.5 为程序运行结果的作图显示，从此图以及以上数值输出结果中可见，病人的体温在 7.56 小时（即早晨 7：34）达到最高值 104.3 °F。读者可以用其他插值阶数重复该插值计算，观察高阶多项式拟合的效果。

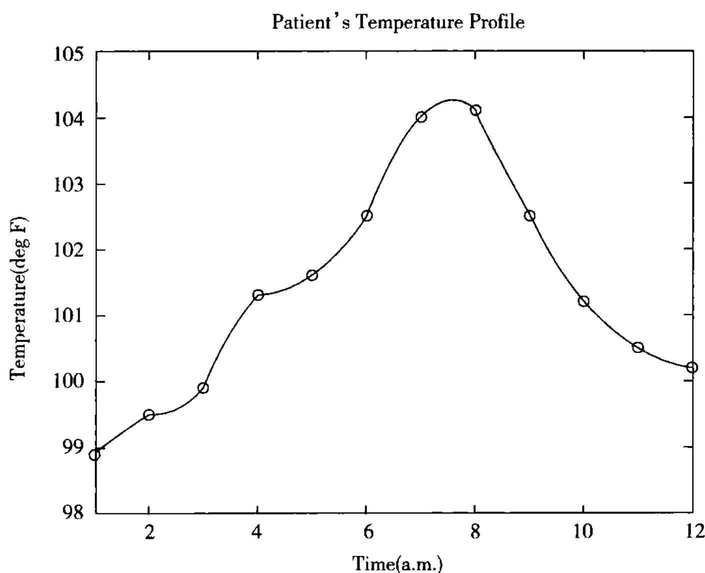


图 6.5　病人体温曲线图

6.8　非等距节点插值

本节将推导非等距节点的两种插值方法，即拉格朗日多项式和样条插值。

6.8.1　拉格朗日多项式

假设已知一组非等距基节点（见图 6.3a），定义如下多项式：

$$P_n(x) = \sum_{k=0}^{n} p_k(x) f(x_k) \tag{6.61}$$

此式是所有（$n+1$）个基节点函数值的加权之和。其中，权重 $p_k(x)$ 是一组 n 次多项式函数，它们以一种特殊的方式与每个基节点相对应。该方程实际上是一个 n 次多项式的线性组合，因此，$P_n(x)$ 也是一个 n 次多项式。

为了使插值多项式在所有基节点处都能完全准确地拟合已知的函数值，每个权重多项式 $p_k(x)$ 必须满足以下条件，即：当 $x = x_k$ 时，$p_k(x)$ 的值为1，而在其他所有基节点处其值为0。也就是：

$$p_k(x_i) = \begin{cases} 0 & i \neq k \\ 1 & i = k \end{cases} \tag{6.62}$$

拉格朗日多项式即为

$$p_k(x) = C_k \prod_{\substack{i=0 \\ i \neq k}}^{n} (x - x_i) \tag{6.63}$$

显然，它满足式（6.62）条件中的第一个等式，因为只要 $x = x_i$，上式的乘积中就有一项 $(x_i - x_i) = 0$。可以设定如下的常数 C_k，使得拉格朗日多项式同时满足式（6.62）条件中的第二个等式，即：

$$C_k = \frac{1}{\prod\limits_{\substack{i=0 \\ i \neq k}}^{n} (x_k - x_i)} \tag{6.64}$$

将此式代入式（6.63），就得到拉格朗日多项式：

$$p_k(x) = \prod_{\substack{i=0 \\ i \neq k}}^{n} \left(\frac{x - x_i}{x_k - x_i} \right) \tag{6.65}$$

该插值多项式 $P_n(x)$ 有一个余项，可以由式（6.6）求得，为

$$R_n(x) = \prod_{i=0}^{n} (x - x_i) \frac{f^{(n+1)}(\xi)}{(n+1)!} \qquad x_0 < \xi < x_n \tag{6.66}$$

6.8.2 样条插值

如图6.6a所示，如果数据节点很多，高次插值多项式很可能在节点之间形成振荡，而不是光滑地连接节点。这时，虽然插值多项式确实是通过所有节点的，但是，用多项式估计节点之间的值并不能令人满意。为了避免这种高次插值多项式的不理想效果，可以用一组低次插值多项式，其中的每个多项式只需拟合几个节点，这组插值多项式称为样条函数。图6.6b所示就是用三次样条对图6.6a数据插值的结果。与高次插值相比，显然，三次样条插值给出了比较好的近似结果。

在工程问题中应用最多的样条函数就是三次样条，该方法将每两个相邻节点

之间的连接曲线用一个三次多项式来近似。由于经过两个点的三次多项式有无穷多个，因此，必须附加限制条件才能得到惟一的样条函数。于是，设定在同一个节点处样条函数具有相等的一、二阶导数值，该条件使得样条多项式在所有节点处具有连续的斜率和曲率。在区间 $[x_{i-1}, x_i]$ 上的三次样条的通式为

$$P_i(x) = a_i x^3 + b_i x^2 + c_i x + d_i \tag{6.67}$$

该方程有 4 个待定系数。在整个插值区间 $[x_0, x_n]$ 上共有 n 个这样的多项式，因此，共有 $4n$ 个待定系数，求解这些系数需要 $4n$ 个方程。这些方程由以下条件给出：

a. 每个样条函数经过各自区间两端的节点，形成 $2n$ 个方程；

b. 样条函数的一阶导数在内节点上连续，形成 $n-1$ 个方程；

c. 样条函数的二阶导数在内节点上连续，形成 $n-1$ 个方程；

d. 整个插值区间两端的样条函数在端节点上的二阶导数为 0，产生 2 个方程。这称为自然边界条件。还有一种常用的条件是使两个端区间的样条函数的三阶导数分别与其相邻区间样条函数的三阶导数相等，这种条件称为非扭结条件。

将这 $4n$ 个线性代数方程联立求解，就可以确定所有三次插值多项式。第 9 章将进一步讨论样条函数的应用。

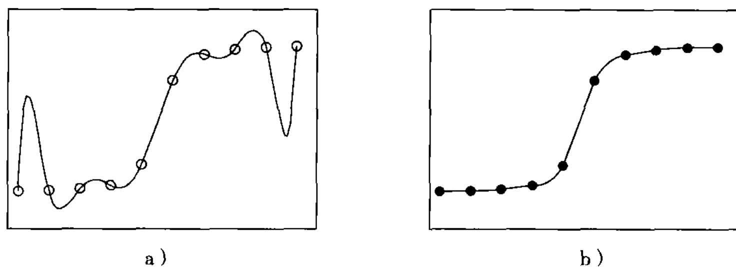


图 6.6

a）高次插值多项式的插值结果，相邻节点之间会出现波动　b）三次样条的插值结果

6.9　积分公式

下面几节将推导积分公式。积分运算

$$I = \int_{x_0}^{x_n} f(x)\,\mathrm{d}x \tag{6.68}$$

是被积函数 $y = f(x)$ 在自变量 x 的积分限 x_0 到 x_n 之间的积分。如果函数 $f(x)$ 的积分存在解析解，就不需要用数值方法。但是，在很多情况下，函数 $f(x)$ 很复杂，或者只是 x 和 y 的一组实验数值表。这时，就需要用数值方法计算式 (6.68) 的积分，这种计算称为数值积分。

如图6.7a所示，由微积分学可知，函数$f(x)$的积分等于积分限范围内函数曲线与x轴之间所围成的面积，其中x轴下面部分的面积为负值（见图6.7b）。于是，求积分$\int_{x_0}^{x_n} y\mathrm{d}x$的一种方法就是画出函数曲线图，测量其围成的面积。但是，这种操作方法不切实际，也很不准确。比较准确并且系统化的求积分方法就是数值积分。下一节，我们将推导等距节点的牛顿-科茨（Newton-Cotes）积分公式。

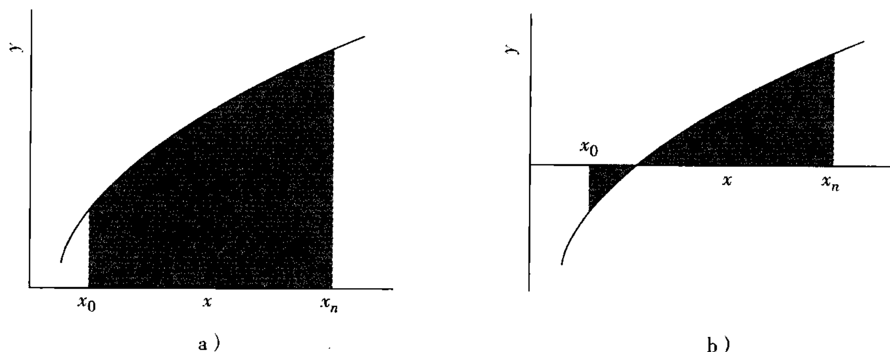


图6.7 积分的示意图
a）只有正值面积 b）有正值面积，也有负值面积

6.10 牛顿-科茨求积公式

这种方法首先用多项式，例如格雷戈里-牛顿向前插值公式（6.57），来近似替代函数$y=f(x)$。实际应用时，把区间$[x_0, x_n]$分段，每段长为h，格雷戈里-牛顿向前插值公式就成为

$$y = y_0 + \frac{(x-x_0)}{h}\Delta y_0 + \frac{(x-x_0)(x-x_1)}{2!h^2}\Delta^2 y_0 + \frac{(x-x_0)(x-x_1)(x-x_2)}{3!h^3}\Delta^3 y_0 + \cdots \tag{6.69}$$

（注意，$x_{i+1}=x_i+h$）。由于该插值公式在$(n+1)$个有限节点处完全拟合函数值，因此，我们把整个积分区间$[x_0,x_n]$分成n段，每段长为h。然后，再用式(6.69)，就可以求出式（6.68）的积分。可以将积分上限增大，则包含的积分段数就增加，每段长仍为h。在式（6.69）序列中保留有限个差分项，并使保留的差分项数等于积分的分段数，这种运算就是著名的牛顿-科茨求积公式。n值为1、2和3时的牛顿-科茨的前3个公式分别称为梯形公式、辛普森1/3公式和辛普森3/8公式。这些公式推导如下。

6.10.1 梯形公式

推导这个牛顿-科茨第一积分公式时，取长为 h 的一个分段（$n = 1$），并用多项式拟合该分段的（x_0, y_0）和（x_1, y_1）两个节点（见图 6.8），取式（6.69）格雷戈里-牛顿多项式的前 2 项（包括一阶向前有限差分项），把其余的项都一起归为余项。这其实就是用直线连接两个节点。于是，积分方程变为

$$I_1 = \int_{x_0}^{x_1}\left[y_0 + \frac{(x - x_0)}{h}\Delta y_0\right]\mathrm{d}x + \int_{x_0}^{x_1} R_n(x)\,\mathrm{d}x \tag{6.70}$$

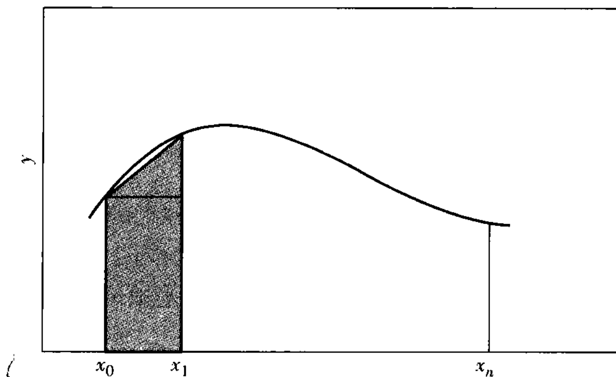


图 6.8　用梯形公式求一个积分段的积分

上式右边的第一个积分是对 x 求积，代入一阶向前差分的定义 $\Delta y_0 = y_1 - y_0$，可得

$$I_1 = \frac{h}{2}(y_0 + y_1) + \int_{x_0}^{x_1} R_n(x)\,\mathrm{d}x \tag{6.71}$$

其中的余项可以用 ξ_1 表示为（推导过程请参见 Constantinides 和 Mostoufi 的著作（1999））

$$\int_{x_0}^{x_1} R_n(x)\,\mathrm{d}x = -\frac{1}{12}h^3 D^2 f(\xi_1) \tag{6.72}$$

这是一个 h^3 阶的项，可以记为 $O(h^3)$。于是，式（6.71）可以写为

$$I_1 = \frac{h}{2}(y_0 + y_1) + O(h^3) \tag{6.73}$$

这个公式就是梯形公式，其中的 $(h/2)(y_0 + y_1)$ 其实是梯形面积的计算公式。如上所述，这种方法拟合两个节点就是用直线连接两节点，如图 6.8 的阴影所示，这使得积分面积的形状成为梯形。曲线 $y = f(x)$ 与 y_0、y_1 两点的连线之间形成的面积就是梯形公式的截断误差。如果 $f(x)$ 原本就是直线，其二阶导数为 0，余项就没有了，梯形公式计算的积分就是准确值。

式（6.73）的梯形公式只给出单个长度为 h 的积分段的积分值。要得到整个曲线 n 段的积分，就必须把式（6.70）用于每个积分段，得到如下一系列方程：

$$I_1 = \frac{h}{2}(y_0 + y_1) + O(h^3) \tag{6.74}$$

$$I_2 = \frac{h}{2}(y_1 + y_2) + O(h^3) \tag{6.75}$$

$$\vdots$$

$$I_n = \frac{h}{2}(y_{n-1} + y_n) + O(h^3) \tag{6.76}$$

I_i 之和就是曲线总积分的近似值。把整个积分区间上的所有这些方程加起来，就形成了复合梯形公式：

$$I = \frac{h}{2}\left(y_0 + 2\sum_{i=1}^{n-1} y_i + y_n\right) + nO(h^3) \tag{6.77}$$

为了简化公式，误差项表示为 $nO(h^3)$。这只是一种近似表示方法，因为每个余项包含了 y 在未知值 ξ_i 处的二阶导数，而每个积分段的 ξ_i 是各不相同的。误差项的绝对值计算不出来，但其相对的大小可以用误差项的阶数表示。由于 n 与 h 成反比，即

$$n = \frac{x_n - x_0}{h} \tag{6.78}$$

复合梯形公式的误差项可以化为

$$nO(h^3) = \frac{x_n - x_0}{h}O(h^3) \cong O(h^2) \tag{6.79}$$

这就是说，在多个积分段上重复应用梯形公式使得计算的准确度降低了大约一个数量级。有关截断误差的较为严格的分析请参见附录 E。

6.10.2 辛普森 1/3 公式

推导这个牛顿-科茨第二求积公式时，要用两个长度都为 h 的积分段（$n=2$）（见图 6.9），多项式拟合 (x_0, y_0)、(x_1, y_1) 和 (x_2, y_2) 3 个节点，也就是用抛物线拟合 3 个点。取式（6.69）格雷戈里-牛顿多项式的前 3 项（包括二阶向前有限差分项），并把其余的项都一起归为余项。积分方程变为

$$I_1 = \int_{x_0}^{x_2}\left[y_0 + \frac{(x-x_0)}{h}\Delta y_0 + \frac{(x-x_0)(x-x_1)}{2!h^2}\Delta^2 y_0\right]\mathrm{d}x + \int_{x_0}^{x_2} R_n(x)\,\mathrm{d}x \tag{6.80}$$

积分该方程，并代入有关有限差分的关系式简化方程，得到

$$I_1 = \frac{h}{3}(y_0 + 4y_1 + y_2) - \frac{1}{90}h^5 D^4 f(\xi_1) \tag{6.81}$$

其误差项为 h^5 阶，可以记为 $O(h^5)$ 。因为只保留了格雷戈里-牛顿多项式中的 3 项，因此，预期的误差项应该为 $O(h^4)$ 阶。不过，余项中 h^4 项的系数为 0，所以，就产生了这么一个碰巧的结果。牛顿-科茨第二公式的最终形式，也就是常说的辛普森 1/3 公式，为

$$I_1 = \frac{h}{3}(y_0 + 4y_1 + y_2) + O(h^5) \tag{6.82}$$

该方程求的是两个积分段上的积分。在每两个积分段上重复应用辛普森 1/3 公式，并把整个积分区间上所有这些方程加起来，就产生了复合辛普森 1/3 公式：

$$I = \frac{h}{3}\left(y_0 + 4\sum_{i=1}^{n/2} y_{2i-1} + 2\sum_{i=1}^{n/2-1} y_{2i} + y_n\right) + O(h^4) \tag{6.83}$$

辛普森 1/3 公式使用成对的积分段，因此，整个积分区间必须分成偶数段。式（6.82）的第一个求和项是奇数项之和，而第二个求和项则是偶数项之和。

与 6.10.1 节所述相同，复合辛普森 1/3 公式的准确度也下降了一个数量级，成为 $O(h^4)$ 。辛普森 1/3 公式的准确度比梯形公式要高，但所需的运算量却增加了。

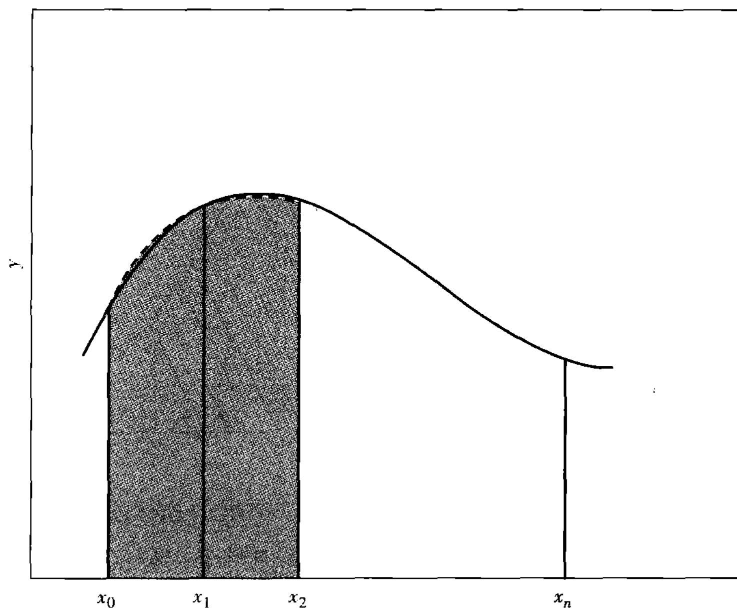


图 6.9　应用辛普森 1/3 公式求两个积分段的积分

6.10.3　辛普森 3/8 公式

推导这个牛顿-科茨第三积分公式时，如图 6.10 所示，要用 3 个长度都为 h 的

积分段($n=3$)，并用多项式拟合(x_0,y_0)、(x_1,y_1)、(x_2,y_2)和(x_3,y_3) 4 个节点，等价于用三次方程拟合这 4 个点。取式（6.69）格雷戈里-牛顿多项式的前 4 项(包括三阶向前有限差分项)，并把其余的项都一起归为余项。积分方程变为

$$I_1=\int_{x_0}^{x_3}\left[y_0+\frac{(x-x_0)}{h}\Delta y_0+\frac{(x-x_0)(x-x_1)}{2!h^2}\Delta^2 y_0+\frac{(x-x_0)(x-x_1)(x-x_2)}{3!h^3}\Delta^3 y_0\right]dx+\int_{x_0}^{x_3}R_n(x)\,dx \tag{6.84}$$

积分该方程，并代入有关有限差分的关系式，简化方程，得到

$$I_1=\frac{3h}{8}(y_0+3y_1+3y_2+y_3)-\frac{3}{80}h^5D^4f(\xi_1) \tag{6.85}$$

其误差项为 h^5 阶，可以记为 $O(h^5)$。该方程的最终形式，也就是常说的辛普森 3/8 公式为

$$I_1=\frac{3h}{8}(y_0+3y_1+3y_2+y_3)+O(h^5) \tag{6.86}$$

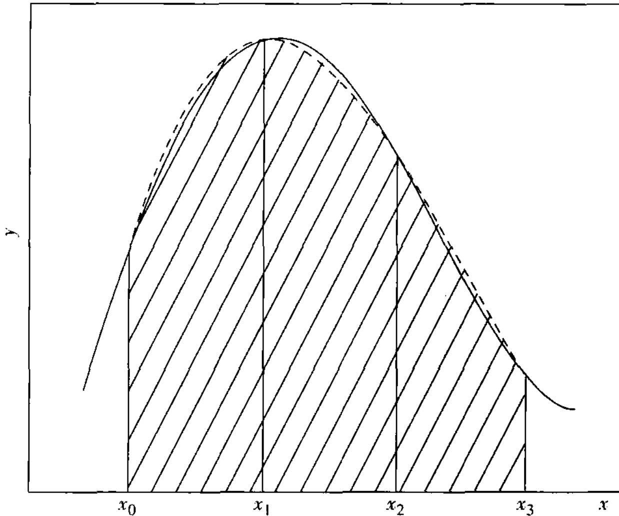


图 6.10 应用辛普森 3/8 公式求 3 个积分段的积分

对于每 3 个积分段重复应用式（6.83），并把整个积分区间上所有这些方程加起来，就得到复合辛普森 3/8 公式：

$$I=\frac{3h}{8}\left(y_0+3\sum_{i=1}^{n/3}(y_{3i-2}+y_{3i-1})+2\sum_{i=1}^{n/3-1}y_{3i}+y_n\right)+O(h^4) \tag{6.87}$$

将辛普森 1/3 公式的误差项与辛普森 3/8 公式的误差项进行比较，可见两者具有同样的阶次，后者只是稍微准确一些。因此，通常辛普森 1/3 公式用得比较多，

因为它只需要 3 个点就可以达到同样阶次的精度，而 3/8 公式却需要 4 个点。

6.10.4　牛顿-科茨求积公式小结

表 6.7 归纳了前几节中推导的 3 个牛顿-科茨求积公式。

在推导牛顿-科茨求积公式的过程中，函数 $y=f(x)$ 用余项为 $R_n(x)$ 的 n 次格雷戈里 - 牛顿多项式 $P_n(x)$ 来近似。其积分计算的通式为

$$\int_a^b y\mathrm{d}x = \int_a^b P_n(x)\mathrm{d}x + \int_a^b R_n(x)\mathrm{d}x \tag{6.88}$$

于是求积公式的一般形式为

$$\int_a^b y\mathrm{d}x = \sum_{i=0}^{n} w_i y_i + O[h^{n+2}, D^{n+1}f(\xi)] \tag{6.89}$$

其中，x_i 为区间 $[a,b]$ 上等距分布的 $(n+1)$ 个节点；权重 w_i 取决于 $P_n(x)$ 多项式对于 $(n+1)$ 个节点的拟合。对于任意阶次小于等于 n 的多项式函数 $y=f(x)$，这个积分是完全准确的，即

$$\int_a^b y\mathrm{d}x = \sum_{i=0}^{n} w_i y_i \tag{6.90}$$

因为对于阶次小于等于 n 的多项式，其导数 $D^{n+1}f(\xi)$ 为 0，于是，误差项 $O[h^{n+2}, D^{n+1}f(\xi)]$ 不存在了。

MATLAB 的 4 个函数 trapz. m、cumtrapz. m、quad. m 和 quad8. m 就是应用不同的牛顿-科茨公式完成某矢量（即函数）的数值求积计算。简介如下：

1）函数 trapz(x,y) 利用梯形公式，计算 y（数据矢量或函数）对 x（变量矢量）的积分。

2）函数 cumtrapz(x,y) 利用梯形公式，计算 y（数据矢量或函数）对 x（变量矢量）的累计积分。

3）函数 quad('file_name',a,b) 利用辛普森 1/3 公式，在区间 $[a,b]$ 上计算 m 文件 file_name. m 表示的函数的积分。

4）函数 quad8('file_name',a,b) 利用 8 个分段（9 个节点）的牛顿-科茨公式，在区间 $[a,b]$ 上计算 m 文件 file_name. m 中存放的函数的积分。

表 6.7　牛顿-科茨数值求积公式小结

公式名称	积分公式	误　差	方程编号
梯形公式	$\int_{x_0}^{x_1} y\mathrm{d}x = \frac{h}{2}(y_0+y_1)$	$-\frac{1}{12}h^3D^2f(\xi)$	(6.91)
辛普森 1/3 公式	$\int_{x_0}^{x_2} y\mathrm{d}x = \frac{h}{3}(y_0+4y_1+y_2)$	$-\frac{1}{90}h^5D^4f(\xi)$	(6.92)

（续）

公式名称	积分公式	误差	方程编号
辛普森 3/8 公式	$\int_{x_0}^{x_3} y\mathrm{d}x = \frac{3h}{8}(y_0 + 3y_1 + 3y_2 + y_3)$	$-\frac{3}{80}h^5 D^4 f(\xi)$	(6.93)
通用求积公式	$\int_{x_0}^{x_n} y\mathrm{d}x = \sum_{i=0}^{n} w_i y_i$	$O[h^{n+2}, D^{n+1}f(\xi)]$	(6.94)

例 6.8 梯形积分公式和辛普森 1/3 积分公式

问题陈述：

二氧化碳释放率和耗氧率是研究发酵过程中微生物生长的两个非常重要的指标，通过分析发酵罐进气口和排气口的气体成分、流速、温度和气体压力，可以算出这两个指标。二氧化碳释放率与耗氧率之比称为呼吸商，可以很好地反映微生物的代谢情况。并且，对二氧化碳释放率和耗氧率求积分，可以得到发酵过程中的二氧化碳的总释放量和总耗氧量。这些总量数据是建立发酵模型物质平衡方程的基础。表 6.8 所示为产黄青霉菌发酵生产青霉素的一组二氧化碳释放率和耗氧率数据。

请编写一个 MATLAB 通用函数，用辛普森 1/3 公式求这组实验数据的积分，并计算在 10h 发酵过程中总的二氧化碳释放量和总耗氧量。将这种算法求得的结果与应用现成 MATLAB 函数 trapz（即梯形公式）求得的结果进行比较。

表 6.8 发酵过程中的二氧化碳释放率和耗氧率数据

发酵时间/h	二氧化碳释放率/（g/h）	耗氧率/（g/h）
140	15.72	15.49
141	15.53	16.16
142	15.19	15.35
143	16.56	15.13
144	16.21	14.20
145	17.39	14.23
146	17.36	14.29
147	17.42	12.74
148	17.60	14.74
149	17.75	13.68
150	18.95	14.51

解题方法：

分别对二氧化碳释放率数据和耗氧率数据求积分。表 6.8 提供了 11 个数据，共形成 10 个数据区间，既可以用梯形公式也可以用辛普森 1/3 公式求积分。按照题目的要求，先用辛普森 1/3 公式，再用梯形公式。

程序说明：

MATLAB 函数 simpson.m 由 Constantinides 和 Mostoufi（1999）开发。函数首先检查输入参数，即自变量矢量 x 和函数矢量 y。两个矢量的长度要相等，x 矢量的元素要等距分布，并且，两个矢量的元素个数应该为奇数，形成偶数个区间。如果矢量包含偶数个元素，形成奇数个区间，那么函数先计算前 $(n-1)$ 个数据点的积分值，再用梯形公式求最后一个数据区间的积分值，然后将两个积分值相加，得到最终的积分值。用户要特别注意这种情况，因为辛普森 1/3 公式截断误差的阶数与梯形公式截断误差的阶数是不同的。完成这些检查之后，函数按照式（6.82）进行积分计算。如果需要的话，按照式（6.75）计算最后一个数据区间的积分值，并将其加入总积分值。

主程序 example6_8.m 运行时，先要求用户通过键盘输入数据，然后调用函数 trapz.m 和 simpson.m 求积分，最后显示结果。

程序如下：

```
% Example6.8.m - Calculates carbon dioxide evolved and
% oxygen uptaken in a fermentation process using TRAPZ
% (trapezoidal rule) and SIMPSON (Simpson's 1/3 rule) functions

clc; clear all

% Input data
t = input(' Vector of time = ');
r_CO2 = input(' Carbon dioxide evolution rate (g/h) = ');
r_O2 = input(' Oxygen uptake rate (g/h) = ');

% Integration
m1CO2 = trapz(t,r_CO2);
m2CO2 = simpson(t,r_CO2);
m1O2 = trapz(t,r_O2);
m2O2 = simpson(t,r_O2);
```

```
% Output
fprintf('\n Total carbon dioxide evolution =% 9.4f (evaluated by
the trapezoidal rule)',m1CO2)
fprintf('\n Total carbon dioxide evolution =% 9.4f (evaluated by
the Simpson 1/3 rule)',m2CO2)
fprintf('\n Total oxygen uptake            =% 9.4f (evaluated by the
trapezoidal rule)',m1O2)
fprintf('\n Total oxygen uptake            =% 9.4f (evaluated by the
Simpson 1/3 rule)\n',m2O2)
```

应用辛普森 1/3 公式求积分的 MATLAB 函数：

```
function Q = Simpson(x , y)
% SIMPSON Numerical evaluation of integral by Simpson's 1/3 rule
%
%   SIMPSON(X,Y) numerically evaluates the integral of the
%   vector of function values Y with respect to X by
%   Simpson's 1/3 rule. X is the vector of equally spaced
%   independent variable. Length of Y has to be odd (even
%   number of intervals). If length of Y is even, the function
%   calculates the integral for [LENGTH(Y) -1] points by
%   Simpson's 1/3 rule and adds to it the value of the
%   integral for the last interval by trapezoidal rule
%   See also TRAPZ , QUAD , QUAD8, GAUSSLEGENDRE

% (c) N. Mostoufi & A. Constantinides
% January 1, 1999

points = length(x);
if length(y) ~= points
   error('x and y are not of the same length')
   break
end

dx = diff(x);
maxi = max([min(abs(x))/1000 , 1e-10]);
```

```
if max(dx) - min(dx) > maxi
    error('X is not equally spaced. ')
    break
end

h = dx(1);
if mod(points,2) == 0
    warning('Odd number of intervals; Trapezoidal rule will be used
for the last interval. ')
    n = points - 1;
else
    n = points;
end

% Integration
y1 = y(2:2:n - 1);
y2 = y(3:2:n - 2);
Q = (y(1) + 4 * sum(y1) + 2 * sum(y2) + y(n)) * h /3;

if n ~= points
    Q = Q + (y(points) + y(n)) * h / 2;
end
```

程序运行时的输入以及输出结果:

```
>> example6_8

Vector of time = [140:150]
Carbon dioxide evolution rate (g/h) = [15.72, 15.53, 15.19,
16.56, 16.21, 17.39, 17.36, 17.42, 17.60, 17.75, 18.95]
Oxygen uptake rate (g/h) = [15.49, 16.16, 15.35, 15.13, 14.20,
14.23, 14.29, 12.74, 14.74, 13.68, 14.51]

Total carbon dioxide evolution  =  168.3450 (evaluated by the
trapezoidal rule)
Total carbon dioxide evolution  =  168.6633 (evaluated by the
```

```
Simpson 1/3 rule)
Total oxygen uptake            =  145.5200 (evaluated by the
trapezoidal rule)
Total oxygen uptake            =  144.9733 (evaluated by the
Simpson 1/3 rule)
```

结果讨论：

表6.9列出了分别利用辛普森1/3公式和梯形公式求实验数据的积分所得到的二氧化碳总释放量和总耗氧量。

表6.9 两种算法的计算结果比较

	辛普森1/3公式	梯形公式
二氧化碳总释放量/g	168.6633	168.3450
总耗氧量/g	144.9733	145.5200

6.11 本章学习要点

学习本章之后，读者应该掌握以下内容：

1）有限差分是建立微分方法和积分方法的主要工具。

2）用有限差分可以表达满足任意准确度要求的导数。

3）利用本章所建立的方法可以计算实验数据或函数的数值积分和数值微分。

4）插值多项式用于建立求积公式。

5）牛顿-科茨求积公式可以用于求实验数据的积分。

6）样条插值常用于非等距数据的插值。

6.12 习题

6.1 请推导向后有限差分项表示的 y 函数的三阶导数公式，并使其误差为 (a) h 阶；(b) h^2 阶。

6.2 请推导向前有限差分项表示的 y 函数的一阶和二阶导数公式，其误差为 h^3 阶。

6.3 请推导格雷戈里-牛顿向后插值公式。

6.4 利用表6.10的实验数据

(a) 建立向前差分表，并用MATLAB函数diff验证结果。

(b) 建立向后差分表。

(c) 应用格雷戈里-牛顿插值公式计算 $x=10$，50，90，130，170和190处的函数值。

表 6.10　青霉素发酵数据

时间/h	青霉素浓度/mL	时间/h	青霉素浓度/mL
0	0	120	9430
20	106	140	10950
40	1600	160	10280
60	3000	180	9620
80	5810	200	9400
100	8600		

6.5　编写 MATLAB 函数，利用格雷戈里-牛顿向后插值公式，由一组 $(n+1)$ 个等距数据求函数 $f(x)$。要求该 MATLAB 函数具有较好的通用性，可以输入任意正整数 n。然后，编写一个 MATLAB 脚本，读入数据，并调用该 MATLAB 函数，考察其拟合数据的结果。用表 6.10 的实验数据验证程序，并计算 $x=10$, 50, 90, 130, 170 和 190 处的函数值。

6.6　对于表 6.11 的非等距数据节点，用拉格朗日多项式和样条插值计算 $x=2$, 4, 5, 8, 9 和 11 处的函数值。

表 6.11　非等距数据

x	$f(x)$	x	$f(x)$
1	7.0	10	8.2
3	3.5	12	9.0
6	3.2	13	9.2
7	3.9		

6.7　为了测定化学反应器的混合特性，在 $t=0$ 时刻向反应器快速注入不参与反应的示踪剂。假设反应器排放物中示踪剂浓度监测值随时间变化的函数为 $c(t)$，则反应器的滞留时间分布 (Residence Time Distribution, RTD) 函数定义为

$$E(t) = \frac{c(t)}{\int_0^{\infty} c(t)\,\mathrm{d}t}$$

累积分布函数定义为

$$F(t) = \int_0^{t} E(t)\,\mathrm{d}t$$

反应器的平均滞留时间为

$$t_m = \frac{V}{q} = \int_0^{\infty} tE(t)\,\mathrm{d}t$$

其中，V 为反应器的体积，q 为流速。则 RTD 函数的方差定义为

$$\sigma^2 = \int_0^{\infty} (t - t_m)E(t)\,\mathrm{d}t$$

在反应器混合特性测定试验中，某个连续流动反应器排放物的示踪剂浓度数据如表 6.12 所示，请计算反应器的 RTD 函数、累积分布函数、平均滞留时间以及 RTD 函数的方差。

表 6.12 化学反应器的示踪剂排放浓度数据

时间/s	$c(t)$/(mg/L)	时间/s	$c(t)$/(mg/L)
0	0	5	5
1	2	6	2
2	4	7	1
3	7	8	0
4	6		

6.13 参考文献

Chapra, S. C., and Canale, R. P. 2006. *Numerical Methods for Engineers*, 4th ed. New York: McGraw-Hill Book Company.

Constantinides, A., and Mostoufi, N. 1999. *Numerical Methods for Chemical Engineers with MATLAB Applications*. Upper Saddle River, NJ: Prentice Hall PTR.

Hanselman, D., and Littlefield, B. 2005. *Mastering MATLAB 7*. Upper Saddle River, NJ: Prentice Hall PTR.

第 7 章　动态系统：常微分方程

7.1　绪论

生理系统的数学模型经常含有常微分方程或偏微分方程，这是因为自然界的生物系统是动态的，它们的状态一直在随着时间或者空间的不同而发生变化。当系统的暂态过程结束之后，模型方程就变成了第 4 章和第 5 章所介绍的代数方程。

本章将讲述常微分方程的数值解。常微分方程用于描述具有单个自变量的生理系统的动态模型，这个自变量可以是空间变量 x 或者是时间变量 t，这取决于系统的结构及其边界条件。由常微分方程描述的生物医学模型有：活细胞的代谢途径、复杂的药物代谢动力学、血氧输送动力学、营养物质在细胞之间的传输、细胞膜和神经细胞动作电位、干细胞的分化和复制、组织细胞的迁移和结合机制、细菌与人群之间相互作用的动态过程等。

本章包括如下学习内容：

1）利用常微分方程建立生理系统动态模型；

2）求解微分方程的数值解，作图表示数值解的结果，并解释生物系统在各种不同条件下的动态行为；

3）评价模型及其数值解的准确度和稳定性。

7.1.1　药物代谢动力学——药物吸收动力学

药物代谢动力学研究药物分布过程以及体内药物浓度的变化速率（Fournier，1999）。药物可以通过胃肠道进入体内，称为肠道给药方式；也可通过其他不同的途径进入体内，如静脉注射、呼吸道吸入、经皮渗透等，这些称为非肠道给药方式。

血液灌注率、毛细血管渗透性、药物的生物亲和力、药物代谢以及肾脏排泄等许多因素都会影响药物在体内的吸收。药物在体内由肝脏内的酶促反应清除，并通过肾脏排泄到尿液中。图 7.1 显示了一个描述药物吸收和清除过程的简化模型，该模型将所有的体液作为一个房室单元看待，它的数学模型是一组线性常微分方程。本章 7.5 节将推导这种方程组的求解方法，并用例 7.3 加以说明。

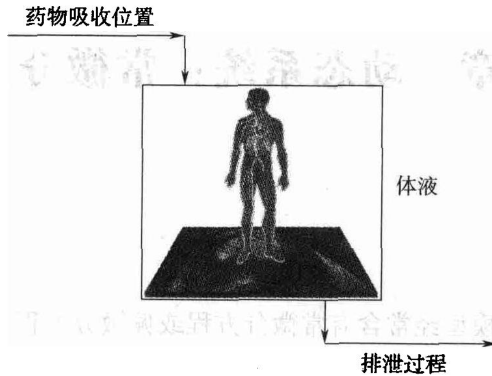


图 7.1 药物吸收的简化模型

7.1.2 组织工程——细胞分化、细胞粘附以及细胞迁移动力学

不同的胚胎细胞或祖细胞特异化为体内具有不同功能的组织，这种动态演变过程称为细胞分化。例如，胎儿体内的胚胎干细胞会复制并分化成为各种类型的细胞，如骨细胞、皮肤细胞、肝细胞、肌细胞等。分化过程包括一系列细胞表型以及形态结构上的变化，尤其在分化后期，这些变化更加明显，很容易直接辨别（Palsson 和 Bhatia，2004）。分化过程从干细胞递交分化信息开始，随后进行一系列有序的基因表达，每一次基因表达都使细胞的分化到达一个新的阶段。通过这一系列不断积累的变化，干细胞最终分化发育成完全成熟的特定细胞群。这些成熟细胞在体内完成它们的既定功能之后，最终要么凋亡，要么通过“转分化”过程转变成其他类型的细胞。干细胞转化成为完全成熟的特异细胞的这一系列过程可以用多房室模型来模拟，根据各个房室的非稳态平衡原理，可以建立模型的常微分方程组。例 7.6 将进一步讨论干细胞分化及其常微分方程组模型的求解过程。

组织工程的研究方向之一就是设计制造合适的多孔基体（即多孔膜），仿真表皮的特性作为康复治疗的辅助支架，用于促进真皮的再生，从而提高创伤和烧伤皮肤的愈合速度。与伤口修复以及组织再生相关的一种细胞活动称为细胞迁移（Lauffenburger 和 Horowitz，1996）。要使细胞在伤口处以及辅助组织再生的植入支架中重新生长，必须有细胞迁移；胚胎形成过程中的细胞分选和器官发育也要有细胞迁移。另外，细胞迁移还与癌症和肿瘤的转移有关。

细胞迁移是特异的细胞表面受体与其配体之间相互作用产生的协同过程，配体通常是胞外基质中的生物分子（见图 7.2）。细胞迁移过程的定量分析需要建立细胞运动（如细胞的运动速度、运动方向、细胞群体的运动特性）与配体的各种特性之间的关系。许多配体特性，如配体的浓度、受体占有率、配体亲和力

等，都会影响细胞的运动。Moghe 和他的同事们（Tjia 和 Moghe，2002a，2002b）建立了一个很有意义的模型，对复杂的细胞迁移过程进行了定量分析。这个细胞迁移模型包括了受配体结合之后配体的细胞内吞过程（根据配体载体的性质，可以是胞吞作用）。他们以动力学原理为基础，利用与常规 Michaelis - Menten 动力学方程相似的扩散反应方程，建立了描述细胞与配体之间相互作用的动力学模型（Tjia 和 Moghe，2002c）。例 7.7 将进一步论述并求解这个描述细胞迁移的数学模型。

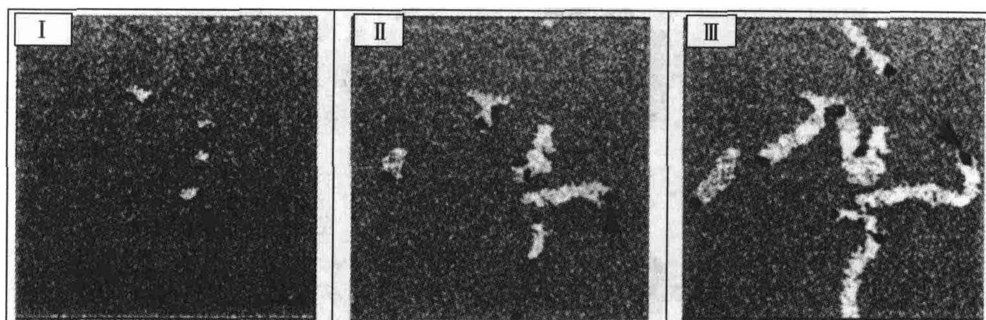


图 7.2　带有配体的微载体促进了皮肤表皮细胞的迁移

图 I 、II 和 III 显示了细胞不断清除配体的轨迹（摘自 Tjia 和 Moghe，2002c）。

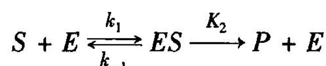
7.1.3　代谢工程——活细胞的糖酵解途径

活细胞分解葡萄糖产生二氧化碳和水的复杂过程称为糖酵解，它由若干酶促反应组成。糖酵解过程中产生的化学能被储存在细胞内的三磷酸腺苷（ATP）化合物中，用于蛋白质等其他化合物的生物合成。糖酵解途径的净反应式为



$$C_6H_{12}O_6 + 6O_2 \rightarrow 6CO_2 + 6H_2O + \text{能量}$$

糖酵解途径中的许多化学反应都需要酶的催化，酶促反应的一般形式为



$$S + E \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} ES \xrightarrow{K_2} P + E$$

其中，酶 E 通过形成中间复合物 ES 催化底物 S 转化为产物 P。酶促反应的动态过程可以用常微分方程组来模拟。本章 7.4 节将推导常微分方程组的求解方法，并将利用这些方法求解例 7.2 中的酶促反应模型。酶促反应的稳态分析可以把微分方程模型简化成为代数方程，用第 4 章和第 5 章讲述的方法就可以求解这些代数方程。

7.1.4　分子的跨膜运输

分子的跨生物膜传输对于活细胞的生理过程至关重要，细胞生长和繁殖所需

的营养物质的供应以及废物从胞内向胞外的传送都涉及跨膜传输。跨膜传输是一个复杂的过程，包含多种机制（见图 7.3）。

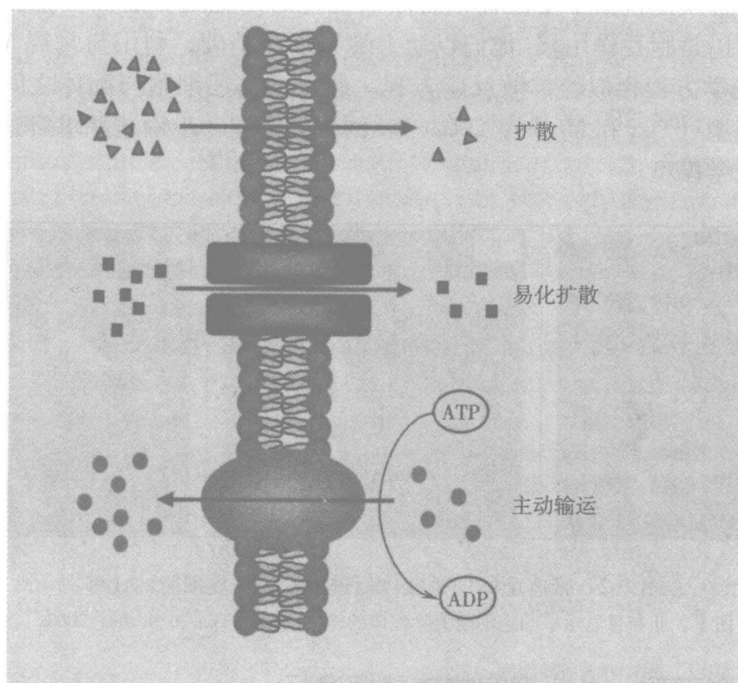


图 7.3 生物膜上的跨膜扩散

分子的被动传输取决于细胞膜内外存在的浓度梯度和电压差的共同作用。中性分子会从高浓度区扩散到低浓度区，带电分子会沿着细胞膜内外存在的电压梯度移动。另外，还有载体介导的传输以及主动传输等产生分子跨膜运动的机制。分子的传输机制可以用常微分方程和偏微分方程描述。本章将讨论可以用常微分方程描述的具有单个自变量的动态传输系统，并在例 7.5 中求解了描述细胞膜和神经细胞动作电位的 Hodgkin-Huxley 模型。在第 8 章我们将讨论可以用偏微分方程描述的两个或更多个自变量的传输系统。

7.2 常微分方程的分类

常微分方程根据方程的阶数、线性与非线性、齐次与非齐次以及边界条件等分类。方程中具有的最高阶导数的阶数就是微分方程的阶。常微分方程分为线性和非线性两类，如果微分方程含有因变量的幂、导数的幂、或者因变量与导数的乘积，则该方程为非线性微分方程。本章中，我们尽可能用符号 y 表示因变量，用符号 t 表示自变量。要记住，通常习惯用 t 或者 x 表示常微分方程的自变量。

n 阶线性常微分方程的一般形式可以写为

$$b_n(t)\frac{d^n y}{dt^n} + b_{n-1}(t)\frac{d^{n-1} y}{dt^{n-1}} + \cdots + b_1(t)\frac{dy}{dt} + b_0(t)y = R(t) \tag{7.1}$$

如果 $R(t)=0$，则方程为齐次方程。如果 $R(t)\neq 0$，则方程为非齐次方程。当系数 $\{b_i \mid i=n, \cdots, 0\}$ 为 t 的函数时，称为变量系数；当系数为常数时，则称为常量系数。以下是一阶、二阶和三阶微分方程的几个例子：

一阶线性齐次微分方程
$$\frac{dy}{dt} + y = 0 \tag{7.2}$$

一阶线性非齐次微分方程
$$\frac{dy}{dt} + y = kt \tag{7.3}$$

一阶非线性非齐次微分方程
$$\frac{dy}{dt} + y^2 = kt \tag{7.4}$$

二阶线性非齐次微分方程
$$\frac{d^2y}{dt^2} + \frac{dy}{dt} + y = e^t \tag{7.5}$$

二阶非线性非齐次微分方程
$$y\frac{d^2y}{dt^2} + \frac{dy}{dt} + y = \cos(t) \tag{7.6}$$

三阶线性齐次微分方程
$$\frac{d^3y}{dt^3} + a\frac{d^2y}{dt^2} + b\frac{dy}{dt} + y = 0 \tag{7.7}$$

三阶非线性非齐次微分方程
$$\frac{d^3y}{dt^3} + a\left(\frac{d^2y}{dt^2}\right)^2 + \frac{dy}{dt} + y = \sin(t) \tag{7.8}$$

式（7.4）、式（7.6）和式（7.8）分别含有 y^2、$y\,(d^2y/dt^2)$ 和 $(d^2y/dt^2)^2$ 项，因此是非线性方程；而式（7.2）、式（7.3）、式（7.5）和式（7.7）则是线性方程。

为了求取 n 阶微分方程或者 n 个一阶微分方程的联立方程组的惟一解，必须已知 n 个特定自变量对应的 n 个因变量或其导数的值，这就是问题求解的初始条件和边界条件。

常微分方程可以分为初值问题和边值问题。所谓初值问题，就是在自变量初始值处，所有因变量和（或者）其导数的值都已知。如果自变量的初始值换成终止值，那么，其实质与初值问题相同，只是积分的方向要反一下，因此，这两种情况都属于初值问题。所谓边值问题，是在多个自变量的点上已知因变量和（或者）其导数的值。如果在自变量初始值处设定某些因变量和（或者）其导数的值，并且，在自变量终止值处设定其他一些因变量和（或者）其导数的值，那么，就称为两点边值问题。

下面的 7.4 节将推导初值问题的求解方法。本书不介绍边值问题的求解方法，有兴趣的读者可以参阅 Constantinides 和 Mostoufi（1999）的著作，以及

Kubícek 和 Hlavàcek（1975）等人的著作。

7.3 标准型的转化

很多常微分方程求积分的方法都要求提供如下形式的 n 个一阶常微分方程组成的联立方程组：

$$\begin{aligned} \frac{dy_1}{dt} &= f_1(t, y_1, y_2, \cdots, y_n) \qquad y_1(t_0) = y_{1,0} \\ \frac{dy_2}{dt} &= f_2(t, y_1, y_2, \cdots, y_n) \qquad y_2(t_0) = y_{2,0} \\ &\vdots \\ \frac{dy_n}{dt} &= f_n(t, y_1, y_2, \cdots, y_n) \qquad y_n(t_0) = y_{n,0} \end{aligned} \tag{7.9}$$

这被称为方程的标准型。如果在某个共同点 t_0 处给定初始条件，则以上方程组的解为

$$\begin{aligned} y_1 &= F_1(t) \\ y_2 &= F_2(t) \\ &\vdots \\ y_n &= F_n(t) \end{aligned} \tag{7.10}$$

利用矩阵符号，可以将该方程组表示为

$$\frac{d\boldsymbol{y}}{dt} = \boldsymbol{f}(t, \boldsymbol{y}) \tag{7.11}$$

初始条件矢量表示为

$$\boldsymbol{y}(t_0) = \boldsymbol{y}_0 \tag{7.12}$$

解矢量表示为

$$\boldsymbol{y} = \boldsymbol{F}(t) \tag{7.13}$$

高阶微分方程或者含有混合阶数的微分方程可以通过一系列代换，转化为标准型。

例如，设 n 阶微分方程为

$$\frac{d^n z}{dt^n} = G\left(z, \frac{dz}{dt}, \frac{d^2 z}{dt^2}, \cdots \frac{d^n z}{dt^n}, t\right) \tag{7.14}$$

将以下转换

$$z = y_1$$

$$\frac{dz}{dt} = \frac{dy_1}{dt} = y_2$$

$$\frac{d^2z}{dt^2}=\frac{dy_2}{dt}=y_3$$
$$\vdots$$
$$\frac{d^{n-1}z}{dt^{n-1}}=\frac{dy_{n-1}}{dt}=y_n \tag{7.15}$$
$$\frac{d^nz}{dt^n}=\frac{dy_n}{dt}$$

代入式（7.14），则得到的等价标准型方程组为如下 n 个一阶方程：

$$\begin{aligned}\frac{dy_1}{dt}&=y_2\\ \frac{dy_2}{dt}&=y_3\\ &\vdots\\ \frac{dy_n}{dt}&=G(y_1,y_2,y_3,\cdots,y_n,t)\end{aligned} \tag{7.16}$$

如果微分方程组的右边都不是自变量的函数，那么这个方程组可以写成：

$$\frac{d\boldsymbol{y}}{dt}=\boldsymbol{f}(\boldsymbol{y}) \tag{7.17}$$

如果函数$\boldsymbol{f}(\boldsymbol{y})$ 是 $\boldsymbol{y}$ 的线性函数，则方程可以写成矩阵形式：

$$\boldsymbol{y}'=\boldsymbol{A}\boldsymbol{y} \tag{7.18}$$

下面的例7.1（a）和（b）就是这样的方程。本书7.4节将讨论非线性方程组的求解方法，7.5节将推导线性常微分方程组的求解方法。

下面举例说明将高阶线性或非线性微分方程转化为标准型的方法。

例 7.1　将常微分方程转化为标准型

问题陈述：

应用式（7.15）和式（7.16）定义的转换公式将以下常微分方程转化为标准型：

（a）
$$\frac{d^4z}{dt^4}+5\frac{d^3z}{dt^3}-2\frac{d^2z}{dt^2}-6\frac{dz}{dt}+3z=0 \quad \text{（是线性齐次方程）}$$

初始条件为

$$t=0\text{ 时},\left.\frac{d^3z}{dt^3}\right|_0=2,\quad \left.\frac{d^2z}{dt^2}\right|_0=1.5,\quad \left.\frac{dz}{dt}\right|_0=1,\quad z|_0=0.5$$

(b)
$$\frac{d^4z}{dt^4}+5\frac{d^3z}{dt^3}-2\frac{d^2z}{dt^2}-6\frac{dz}{dt}+3z=e^{-t}\quad(\text{是线性非齐次方程})$$

初始条件为

$$t=0\text{ 时},\left.\frac{d^3z}{dt^3}\right|_0=2,\quad \left.\frac{d^2z}{dt^2}\right|_0=1.5,\quad \left.\frac{dz}{dt}\right|_0=1,\quad z|_0=0.5$$

(c)
$$\frac{d^3z}{dt^3}+z^2\frac{d^2z}{dt^2}-\left(\frac{dz}{dt}\right)^3-2z=0\qquad(\text{是非线性齐次方程})$$

边界条件为

$$t=0\text{ 时},\left.\frac{d^2z}{dt^2}\right|_0=1,\quad \left.\frac{dz}{dt}\right|_0=2,\quad z|_0=3$$

解：

(a) 应用式 (7.15) 转换公式，得到以下 4 个方程：

$$\frac{dy_1}{dt}=y_2\qquad\qquad y_1(0)=0.5$$

$$\frac{dy_2}{dt}=y_3\qquad\qquad y_2(0)=1$$

$$\frac{dy_3}{dt}=y_4\qquad\qquad y_3(0)=1.5$$

$$\frac{dy_4}{dt}=-3y_1+6y_2+2y_3-5y_4\qquad\qquad y_4(0)=2$$

这是一个线性常微分方程组，可以表示为式 (7.18) 的矩阵形式，其中矩阵 $\boldsymbol{A}$ 为

$$\boldsymbol{A}=\begin{bmatrix}0&1&0&0\\0&0&1&0\\0&0&0&1\\-3&6&2&-5\end{bmatrix}$$

线性常微分方程组的求解方法在 7.5 节讨论。

(b) 此方程右边有 e^{-t}项，所以是非齐次方程。方程的左边与方程 (a) 相同，因此，就用方程 (a) 的转换式。要替换 e^{-t}项，还需要加上如下转换式：

$$y_5=e^{-t}$$

$$\frac{dy_5}{dt}=-e^{-t}=-y_5$$

将这些转换式代入方程 (b)，就得到以下 5 个线性常微分方程：

$$\frac{dy_1}{dt} = y_2 \qquad y_1(0) = 0.5$$

$$\frac{dy_2}{dt} = y_3 \qquad y_2(0) = 1$$

$$\frac{dy_3}{dt} = y_4 \qquad y_3(0) = 1.5$$

$$\frac{dy_4}{dt} = -3y_1 + 6y_2 + 2y_3 - 5y_4 + y_5 \qquad y_4(0) = 2$$

$$\frac{dy_5}{dt} = -y_5 \qquad y_5(0) = 1$$

该方程组是线性的，可以简化为式（7.18）的矩阵形式，其中矩阵 $\boldsymbol{A}$ 为

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ -3 & 6 & 2 & -5 & 1 \\ 0 & 0 & 0 & 0 & -1 \end{bmatrix}$$

（c）该方程是非线性的，但是，可以用如下相似的转换式：

$$z = y_1$$

$$\frac{dz}{dt} = \frac{dy_1}{dt} = y_2$$

$$\frac{d^2z}{dt^2} = \frac{dy_2}{dt} = y_3$$

$$\frac{d^3z}{dt^3} = \frac{dy_3}{dt}$$

代入方程（c），得到

$$\frac{dy_1}{dt} = y_2 \qquad y_1(0) = 3$$

$$\frac{dy_2}{dt} = y_3 \qquad y_2(0) = 2$$

$$\frac{dy_3}{dt} = 2y_1 + y_2^3 - y_1^2 y_3 \qquad y_3(0) = 1$$

显然，这是一个非线性微分方程组，不能表示为矩阵形式。下一节将推导非线性微分方程的求解方法。

7.4　非线性常微分方程组

本节推导标准型常微分方程组数值解的求法。常微分方程组的标准型为

$$\frac{d\boldsymbol{y}}{dt} = \boldsymbol{f}(t,\boldsymbol{y}) \tag{7.11}$$

其初始条件矢量为

$$\boldsymbol{y}(t_0) = \boldsymbol{y}_0 \tag{7.12}$$

为了能够作图说明求解方法，我们先把 y 看作单变量，而不是矢量变量，也就是先考虑单个微分方程。由单个微分方程推导出来的求解公式可以推广到微分方程组的求解。7.4.3 节会证明这一点。

下面开始推导求解方法。首先，重排式（7.11），两边求积分，积分区间为 $t_i \leqslant t \leqslant t_{i+1}$ 和 $y_i \leqslant y \leqslant y_{i+1}$ ：

$$\int_{y_i}^{y_{i+1}} dy = \int_{t_i}^{t_{i+1}} f(t,y)\,dt \tag{7.19}$$

可以求得左边的积分为

$$y_{i+1} - y_i = \int_{t_i}^{t_{i+1}} f(t,y)\,dt \tag{7.20}$$

这个积分的一种求法就是取方程的左边，用泰勒级数求其近似值。这种方法直接采用因变量 y 的切线斜率，而不是求函数 $f(t,y)$ 下面的面积。下面的 7.4.1 节和 7.4.2 节就讲述这种求解方法。

7.4.1　欧拉法和改进欧拉法

欧拉法是最早建立的常微分方程求解方法之一，它来源于式（6.27）：

$$\frac{dy_i}{dx} = \frac{1}{h}(y_{i+1} - y_i) + O(h) \tag{6.27}$$

从例 6.3 的推导过程已知，该式来自于 y_{i+1} 相对于 y_i 的泰勒级数展开式。重排这个方程，可以得到 y 值的向前计算公式：

$$y_{i+1} = y_i + h\frac{dy_i}{dt} + O(h^2) \tag{7.21}$$

这就是微分方程求积的显式欧拉公式。“显式”的意思是方程中只含有位于左边的一个未知量 y_{i+1}，其值可以用方程右边的已知值求得。用 y_i' 或 $f(t_i, y_i)$ 代替式中的导数，得到如下更常用的显式欧拉公式：

$$y_{i+1} = y_i + hf(t_i,y_i) + O(h^2) \tag{7.22}$$

以后，y'_i 与 $f(t_i, y_i)$ 等价使用。应该记住，这两项之所以相等，是因为存在式（7.11）这个微分方程。

式（7.22）显式欧拉法的含义是：下一个 y 值可以由前一个值向 y 的切线方向移动一个步长 h 得到。示意图 7.4a 说明了这一点。这种欧拉公式具有二阶截断误差 $O(h^2)$，很不准确。如图 7.4b 所示，如果 h 较大，y 的轨迹很快就会偏离其真实值。

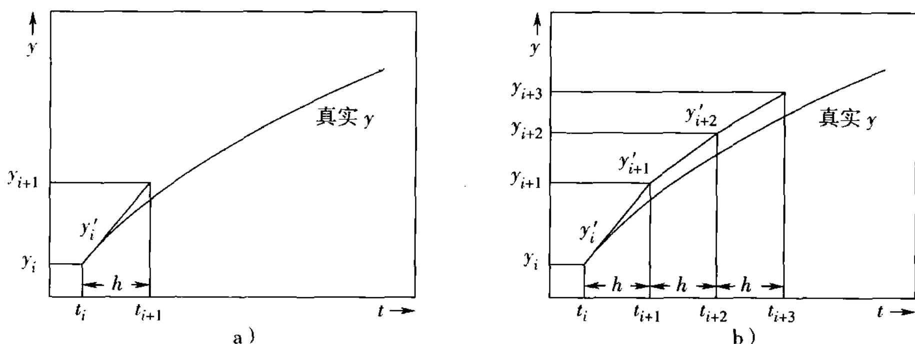


图 7.4　显式欧拉积分法

a）单步　b）多步

用向前泰勒级数和向后泰勒级数相结合的方法可以改善欧拉法的准确度。下面就从式（6.14）开始，该式由向后泰勒级数和向后有限差分得到（见例 6.1）：

$$\frac{dy_i}{dx} = \frac{1}{h}(y_i - y_{i-1}) + O(h) \tag{6.14}$$

并按照此公式写出（$i+1$）处的导数：

$$\frac{dy_{i+1}}{dx} = \frac{1}{h}(y_{i+1} - y_i) + O(h) \tag{7.23}$$

重排此式，求解 y_{i+1}，并用 $f(t_{i+1}, y_{i+1})$ 代替导数，得到

$$y_{i+1} = y_i + hf(t_{i+1}, y_{i+1}) + O(h^2) \tag{7.24}$$

这就是隐式欧拉公式（也就是向后欧拉公式），需要计算函数 f 在未知数 y_{i+1} 上的值。该式的含义是：从 i 位置向前移动一步到 $i+1$ 位置时所取的方向必须由 $i+1$ 处的计算值决定。

隐式方程不能单独求解，必须组成代数方程组才能求解。如果方程组是线性的，则可以用第 4 章讲述的高斯消元法求解。如果方程组中含有非线性方程，问题就难得多，必须用第 5 章讲述的非线性代数方程组的牛顿法求解。

对于隐式欧拉公式，可以先用显式欧拉法预测一个 y_{i+1} 的值，以此简化问题的求解：

$$(y_{i+1})_{\text{预测}} = y_i + hf(t_i, y_i) + O(h^2) \tag{7.25}$$

然后再把这个预测值代入式（7.24）隐式欧拉公式，得到一个校正值：

$$(y_{i+1})_{校正} = y_i + hf[t_{i+1}, (y_{i+1})_{预测}] + O(h^2) \tag{7.26}$$

这两步的结合称为欧拉预测-校正，也就是改进的欧拉法。式（7.26）的校正可以反复使用，直到校正值收敛为止，也就是两个连续的校正值之差小于设定的收敛指标。但是，第二次以后的校正对于准确度的提高不再会有很大的作用。

显式欧拉法和隐式欧拉法的误差都是（h^2）阶。不过，结合预测-校正法之后，准确度提高，误差变为（h^3）阶。证明如下：将 $y_{i+1} = y_i + \Delta y_i$ 和 $y_{i+1} = y_i + \nabla y_{i+1}$ 两式相加，并重排，得到

$$y_{i+1} = y_i + \frac{1}{2}(\Delta y_i + \nabla y_{i+1}) \tag{7.27}$$

利用泰勒级数展开，得到

$$y_{i+1} = y_i + \frac{h}{2}[f(t_i, y_i) + f(t_{i+1}, y_{i+1})] + O(h^3) \tag{7.28}$$

由于（h^2）项的符号正好相反，因此被抵消，从而提高了公式的准确度。

将式（7.28）改写为

$$y_{i+1} = y_i + \frac{h}{2}f(t_i, y_i) + \frac{h}{2}f(t_{i+1}, y_{i+1}) + O(h^3) \tag{7.29}$$

可见这种方法是采用了步长 h 两个端点上 y 函数斜率的平均值（即两个斜率的权重相等，均为 1/2）。这个公式也称为克拉克-尼科尔森（Crank-Nicolson）法。

式（7.29）可以写成如下更一般的形式：

$$y_{i+1} = y_i + w_1 k_1 + w_2 k_2 \tag{7.30}$$

其中

$$k_1 = hf(t_i, y_i) \tag{7.31}$$

$$k_2 = hf(t_i + c_2 h, y_i + a_{21} k_1) \tag{7.32}$$

如何选取权重因子 w_1 和 w_2 以及计算斜率的位置 i 和（$i+1$），取决于积分公式所要求的准确度，也就是，由无限级数展开中保留的项数决定。

这就是接下来要讨论的一系列常微分方程积分公式的基础，这些积分公式具有不同的准确度。

7.4.2 龙格-库塔法

最常用的常微分方程求积方法有一个系列，它们是二阶、三阶、四阶和五阶龙格-库塔法，加上其他几个由龙格-库塔法演变而来的方法。这些方法都基于上一节（7.4.1 节）末尾提到的加权斜率公式。求解（7.11）微分方程的向前积分公式的一般形式为

$$y_{i+1} = y_i + w_1 k_1 + w_2 k_2 + w_3 k_3 + \cdots + w_m k_m \tag{7.33}$$

这是一个迭代公式，其中，各个斜率 k_i 的计算如下：

$$
\begin{aligned}
k_1 &= hf(t_i, y_i) \\
k_2 &= hf(t_i + c_2 h, y_i + a_{21} k_1) \\
k_3 &= hf(x_i + c_3 h, y_i + a_{31} k_1 + a_{32} k_2) \\
&\vdots \\
k_m &= hf(x_i + c_m h, y_i + a_{m1} k_1 + a_{m2} k_2 + \cdots + a_{m,m-1} k_{m-1})
\end{aligned}
\tag{7.34}
$$

这组公式可以简写为

$$y_{i+1} = y_i + \sum_{j=1}^{m} w_j k_j \tag{7.35}$$

$$k_j = hf\left(x_i + c_j h, y_i + \sum_{l=1}^{t-1} a_{jl} k_l\right) \tag{7.36}$$

其中，$c_1 = 0$，$a_{1j} = 0$。y_{i+1}的无限展开级数中保留了 $m+1$ 项，m 的值就决定了算法的复杂度和准确度。

本书不介绍龙格-库塔法的推导过程，有兴趣的读者可以参阅 Constantinides 和 Mostoufi（1999）的著作，该书详细推导了二阶龙格-库塔法。

表 7.1 列出了几个龙格-库塔公式。四阶龙格-库塔法可能是用得最多的常微分方程数值求积方法，其误差为 $O(h^5)$。隐式龙格-库塔法比显式方法的稳定区域要宽，Hairer 等人对这个问题作了详细的论述（Hairer 等，1980，Hairer 和 Wanner，1991a，1991b）。具有步长控制的五阶隐式龙格-库塔法，比如 Radau5 算法，可用于刚性微分方程组的求解。本书不深入讨论这些问题，有兴趣的读者可以参阅上述参考文献。

表 7.1 龙格-库塔积分公式小结

二阶龙格-库塔法（也就是克拉克-尼科尔森法）

$$
\begin{aligned}
y_{i+1} &= y_i + \frac{1}{2}(k_1 + k_2) + O(h^3) \\
k_1 &= hf(t_i, y_i) \\
k_2 &= hf(t_i + h, y_i + k_1)
\end{aligned}
\tag{7.37}
$$

三阶龙格-库塔法

$$
\begin{aligned}
y_{i+1} &= y_i + \frac{1}{6}(k_1 + 4k_2 + k_3) + O(h^4) \\
k_1 &= hf(t_i, y_i) \\
k_2 &= hf\left(t_i + \frac{h}{2}, y_i + \frac{k_1}{2}\right) \\
k_3 &= hf(t_i + h, y_i + 2k_2 - k_1)
\end{aligned}
\tag{7.38}
$$

（续）

四阶龙格-库塔法	$$\begin{aligned} y_{i+1} &= y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) + O(h^5) \\ k_1 &= hf(t_i, y_i) \\ k_2 &= hf(t_i + \frac{h}{2}, y_i + \frac{k_1}{2}) \\ k_3 &= hf(t_i + \frac{h}{2}, y_i + \frac{k_2}{2}) \\ k_4 &= hf(t_i + h, y_i + k_3) \end{aligned} \quad (7.39)$$

7.4.3 微分方程组

在7.4节的开头曾经提到过单个微分方程的求解方法可以推广用于微分方程组的求解。为了说明这一点，假设有如下规范型的 n 个联立常微分方程：

$$\begin{aligned} \frac{dy_1}{dt} &= f_1(t, y_1, y_2, \cdots, y_n) \\ \frac{dy_2}{dt} &= f_2(t, y_1, y_2, \cdots, y_n) \\ &\vdots \\ \frac{dy_n}{dt} &= f_n(t, y_1, y_2, \cdots, y_n) \end{aligned} \quad (7.40)$$

比如采用四阶龙格-库塔公式求该方程组的解，则有

$$\begin{aligned} y_{i+1,j} &= y_i + \frac{1}{6}(k_{1j} + 2k_{2j} + 2k_{3j} + k_{4j}) + O(h^5) && j = 1,2,\cdots,n \\ k_{1j} &= hf_j(t_i, y_{i1}, y_{i2}, \cdots, y_{in}) && j = 1,2,\cdots,n \\ K_{2j} &= hf_j\left(t_i + \frac{h}{2}, y_{i1} + \frac{k_{11}}{2}, y_{i2} + \frac{k_{12}}{2}, \cdots, y_{in} + \frac{k_{1n}}{2}\right) && j = 1,2,\cdots,n \\ k_{3j} &= hf_j\left(t_i + \frac{h}{2}, y_{i1} + \frac{k_{21}}{2}, y_{i2} + \frac{k_{22}}{2}, \cdots, y_{in} + \frac{k_{2n}}{2}\right) && j = 1,2,\cdots,n \\ k_{4j} &= hf_j(t_i + h, y_{i1} + k_{31}, y_{i2} + k_{32}, \cdots, y_{in} + k_{3n}) && j = 1,2,\cdots,n \end{aligned} \quad (7.41)$$

利用嵌套循环可以编程实现这个算法。在 MATLAB 中，k 的值和 y_i 的值可以用矢量表示，于是，式（7.41）就很容易用矩阵形式求解。

7.4.4 求解非线性微分方程组的 MATLAB 函数

MATLAB 有几个函数可用于求解式（7.11）形式的常微分方程组，表7.2列出了这些解法器以及它们所采用的求解方法。下面这些语句都可以用于调用常微分方程

(ODE) 解法器：

```
[T,Y] = solver(@name_func, tspan, y0)
[T,Y] = solver(@name_func, tspan, y0, options)
[T,Y] = solver(@name_func, tspan, y0, options, p1, p2,...)
```

其中，“solver”就是 ode23、ode45、ode113、ode15s、ode23s、ode23t 和 ode23tb 这些解法器之一。解法器的输入参数有：

name_func：是包含微分方程组等式右边部分的 m 文件的文件名，该 m 文件是个函数，调用 name_func(t,y) 函数返回一个列矢量，它对应于微分方程的 $\boldsymbol{f}(t,\boldsymbol{y})$。

tspan：这是一个矢量，它定义了积分区间 [t0, tf]。若要计算单调递增或单调递减的一组特定点上的值，用 tspan = [t0, t1, …, tf] 表示；若要计算等距分布点上的值，用 tspan = [t0:delt:tf] 表示，其中 delt 是用户设定的邻近两点之间的距离。

y0：是包含微分方程初始条件的一个矢量。

options：这是用 odeset 函数建立的积分参数。在 MATLAB 指令窗中键入 help odeset 指令可以查看有关该函数的详细说明。

p1，p2，…：这是解法器传递给 name_func 函数和“options”中设定的其他函数的参数。

[T, Y]：解法器返回的自变量和因变量的值分别存放在矢量 T 和 Y 中。如果用户没有用上述方法设定 tspan，积分解法器就会自动控制步长，使得自变量矢量不是等距分布的。

表 7.2 MATLAB 的常微分方程（ODE）解法器

解法器	求解方法
ode23	低阶龙格-库塔法（二阶、三阶段）
ode45	高阶龙格-库塔法（四阶、五阶段）
ode113	阶数可变（1～13 阶）的亚当斯-巴什福思-莫尔顿法
ode15s	隐式、多步、阶数可变（1～5 阶），适用于刚性微分方程组
ode23s	二阶改进 Rosenbrock 方法，适用于刚性微分方程组
ode23t	梯形公式的一种“自由”插值实现，适用于求解中度刚性微分方程组
ode23tb	一种隐式龙格-库塔公式的实现，具有两个阶段。第一阶段是梯形公式，第二阶段是二阶向后微分公式，适用于刚性微分方程组

例如：

```
[T,Y] = ode45(@test1_func,[0:10],[1,0],[],0.1,0.02,0.1)
function dydt = test1_func(t,y,p1,p2,p3)
dydt = [p1 * y(1) - p2 * y(2)^2;p3 * exp(y(1))];
```

该 MATLAB 函数 test1 _ func 用一个列矢量 dydt 返回导数的值。即使自变量 t 没有显式出现在导数的定义中，函数的第一个输入参数也必须是 t。函数的第二个输入参数是因变量矢量 y。附加参数 p1、p2 和 p3 是 ode45 调用中的最后 3 个值，即 0.1、0.02 和 0.1，被传递给 test1 _ func 函数。

函数调用的另一种方法是：

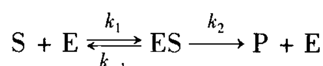
```
[T,Y] = ode45('test2 _ func',[0:10],[1,0],[],0.1,0.02,0.1)
function dydt = test2 _ func(t,y,flag,p1,p2,p3)
dydt = [p1 * y(1) - p2 * y(2)^2;p3 * exp(y(1))];
```

注意，此处 test2 _ func 函数的第 3 个输入参数 flag 是空变量，从第 4 个参数开始才是附加参数。

例 7.2 酶促反应的求解

问题陈述

假设有一种酶 E，通过与底物 S 形成中间产物 ES 来催化底物转化成产物 P，其反应式为



$$S + E \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} ES \xrightarrow{k_2} P + E$$

应用物质作用定律可以建立描述该酶促反应动力学过程的微分方程组。请利用以下初始条件和速率常数，求解微分方程组，并作出模型中各个变量随时间变化的曲线。

初始条件： $[S]_0 = 1.0\mu M$ $[E]_0 = 0.1\mu M$ $[ES]_0 = 0$ $[P]_0 = 0$

常量： $k_1 = 0.1(\mu M)^{-1}s^{-1}$ $k_{-1} = 0.1s^{-1}$ $k_2 = 0.3s^{-1}$

同时确定反应达到 99.9% 底物转化率所需要的时间（精确到 s）。

解：

根据物质作用定律，稀释气体或溶液中，两个反应物分子的碰撞速率与两个反应物浓度的乘积成正比。因此，可以建立如下模型方程：

$$\frac{d[S]}{dt} = -k_1[S][E] + k_{-1}[ES] \qquad [S]_0 = 1.0$$

$$\frac{d[E]}{dt} = -k_1[S][E] + k_{-1}[ES] + k_2[ES] \qquad [E]_0 = 0.1$$

$$\frac{d[ES]}{dt} = k_1[S][E] - k_{-1}[ES] - k_2[ES] \qquad [ES]_0 = 0$$

$$\frac{d[P]}{dt} = k_2[ES] \qquad [P]_0 = 0$$

用下列程序 example7 _ 2. m 和函数 enzyme _ kinetics _ equations. m 求方程组在 0 ~ 1000s 时间范围内的积分。

MATLAB 程序为

```
% example7_2.m - Integration of simple enzyme kinetics model
% using MATLAB function ode45.m to integrate the differential
% equations that are contained in the file:
% enzyme_kinetics_equations.m

clc; clear all;
% Set the initial conditions, constants, & time span
yzero = [1, 0.1, 0, 0];
k1 = 0.1; k_1 = 0.1; k2 = 0.3;
tspan = [0 1000];

% Integrate the equations
[t,y] = ode45(@enzyme_kinetics_equations,tspan,yzero,[],k1,
k_1,k2);
n = length(t);
% Print out the results
n = length(y);
for i = 1:n
if y(i,1) <= 0.001 * yzero(1)
fprintf('Reaction is 99.9 percent complete at time = % 4.0f sec-
onds',t(i));
break
end
end
% Plot concentration profiles
clf; figure(1); plot(t,y(:,1),'-',t,y(:,4),'-.')
title('Concentration Profiles of Substrate and Product','FontSize
',12)
xlabel('Time, s','FontSize',12);
ylabel('Concentration, \muM','FontSize',12);
legend('S','P');
figure(2); plot(t,y(:,2),'-',t,y(:,3),'-.')
title('Concentration Profiles of Enzyme and Complex','FontSize',12)
xlabel('Time, s','FontSize',12);
ylabel('Concentration, \muM','FontSize',12);
legend('E','ES')
```

包含微分方程组的函数为

```
function dy = enzyme_kinetics_equations(t,y,k1,k_1,k2)
% enzyme_kinetics_equations.m
% Contains the equations for example7_2

% Variables
S = y(1); E = y(2); ES = y(3);
% Equations
dy = [ - k1 * S * E + k_1 * ES
 - k1 * S * E + k_1 * ES + k2 * ES
k1 * S * E - k_1 * ES - k2 * ES
k2 * ES];
```

求解结果:

如图7.5a和图7.5b的曲线所示，复合物［ES］在反应的初始几秒钟时间内迅速生成，底物S逐渐转化为产物P。程序运行的如下输出结果显示在960s时99.9%的底物已转化成为产物，此时，酶的复合物没有了，酶还原为原来的游离状态。

```
Reaction is 99.9 percent complete at time = 960 seconds
```

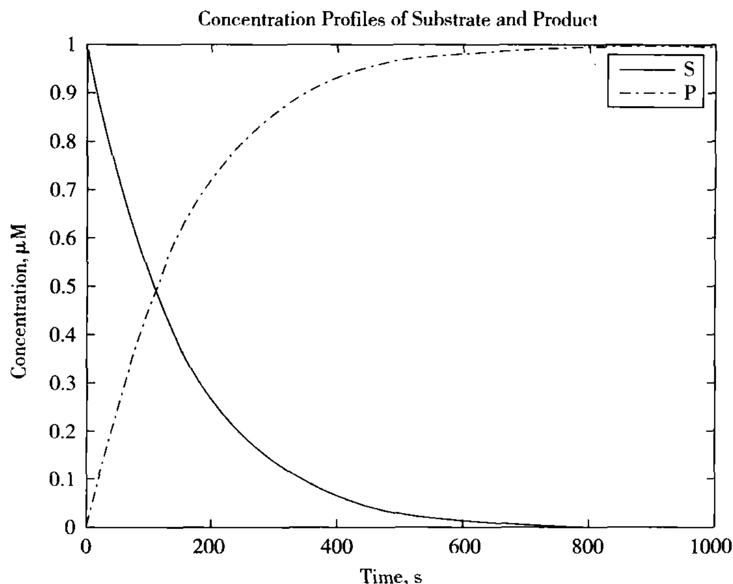


图7.5a　底物和产物的浓度曲线

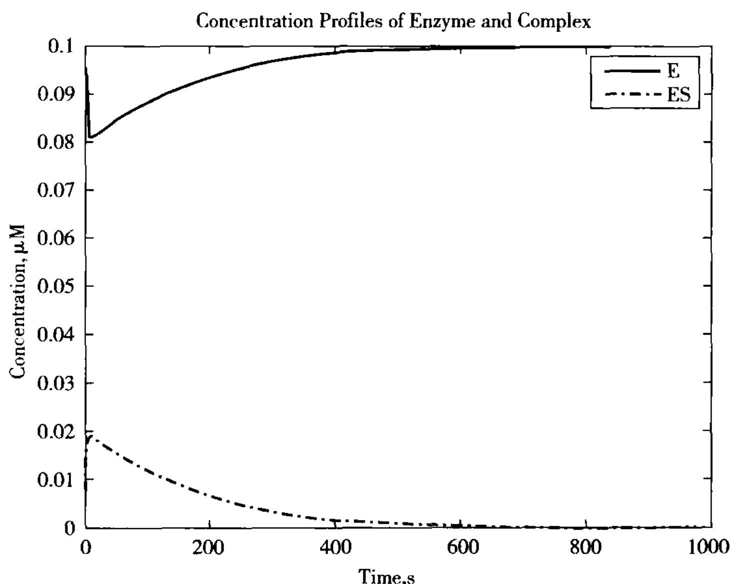


图 7.5b　酶及其复合物的浓度曲线

7.5　线性常微分方程组

很多生物医学系统的数学模型是常系数线性常微分方程组，可以简写为如下形式：

$$\boldsymbol{y}' = \boldsymbol{A}\boldsymbol{y} \tag{7.18}$$

给定的初始条件为

$$\boldsymbol{y}(0) = \boldsymbol{y}_0 \tag{7.42}$$

7.5.1　应用特征值和特征矢量的求解方法

常系数线性常微分方程组具有特定形式的解，可以由矩阵 $\boldsymbol{A}$ 的特征值和特征矢量求得。为了推导这个解，我们先考虑如下单个线性微分方程：

$$\frac{\mathrm{d}y}{\mathrm{d}t} = ay \tag{7.43}$$

其初始条件为

$$y(0) = y_0 \tag{7.44}$$

式（7.43）其实是式（7.18）矩阵方程的标量形式。通过分离变量并将方程两边求积分，可以求得该标量方程的解：

$$\int_{y_0}^{y}\frac{\mathrm{d}y}{y}=\int_0^t a\mathrm{d}t$$

$$\ln\frac{y}{y_0}=at \tag{7.45}$$

$$y=\mathrm{e}^{at}y_0$$

同理，矩阵方程的解为

$$\boldsymbol{y}=\mathrm{e}^{\boldsymbol{A}t}\boldsymbol{y}_0 \tag{7.46}$$

式中 $\boldsymbol{y}$，$\boldsymbol{y}_0$——因变量矢量和初始条件矢量；

$\mathrm{e}^{\boldsymbol{A}t}$——矩阵指数函数，可用下式求值：

$$\mathrm{e}^{\boldsymbol{A}t}=\boldsymbol{I}+\boldsymbol{A}t+\frac{\boldsymbol{A}^2t^2}{2!}+\frac{\boldsymbol{A}^3t^3}{3!}+\frac{\boldsymbol{A}^4t^4}{4!}+\cdots \tag{7.47}$$

将式（7.46）求导，可以证明它就是式（7.18）方程组的解：

$$\begin{aligned}\frac{\mathrm{d}\boldsymbol{y}}{\mathrm{d}t}&=\frac{\mathrm{d}}{\mathrm{d}t}(\mathrm{e}^{\boldsymbol{A}t})\boldsymbol{y}_0\\&=\frac{\mathrm{d}}{\mathrm{d}t}\left(\boldsymbol{I}+\boldsymbol{A}t+\frac{\boldsymbol{A}^2t^2}{2!}+\frac{\boldsymbol{A}^3t^3}{3!}+\frac{\boldsymbol{A}^4t^4}{4!}+\cdots\right)\boldsymbol{y}_0\\&=\left(\boldsymbol{A}+\boldsymbol{A}^2t+\frac{\boldsymbol{A}^3t^2}{2!}+\frac{\boldsymbol{A}^4t^3}{3!}+\cdots\right)\boldsymbol{y}_0\\&=\boldsymbol{A}\left(\boldsymbol{I}+\boldsymbol{A}t+\frac{\boldsymbol{A}^2t^2}{2!}+\frac{\boldsymbol{A}^3t^3}{3!}+\cdots\right)\boldsymbol{y}_0\\&=\boldsymbol{A}(\mathrm{e}^{\boldsymbol{A}t})\boldsymbol{y}_0\\&=\boldsymbol{A}y\end{aligned} \tag{7.48}$$

因为需要计算指数 $\mathrm{e}^{\boldsymbol{A}t}$项的无穷级数，所以，通过计算式（7.47）求解线性常微分方程组是很困难的。但是，进一步应用代数运算，可以用矩阵 $\boldsymbol{A}$ 的特征值和特征矢量表示这个解。由附录 C 可知，n 阶非奇异矩阵 $\boldsymbol{A}$ 有 n 个特征矢量和 n 个非 0 特征值，定义为

$$\begin{aligned}\boldsymbol{A}\boldsymbol{x}_1&=\lambda_1\boldsymbol{x}_1\\\boldsymbol{A}\boldsymbol{x}_2&=\lambda_2\boldsymbol{x}_2\\&\vdots\\\boldsymbol{A}\boldsymbol{x}_n&=\lambda_n\boldsymbol{x}_n\end{aligned} \tag{7.49}$$

所有这些特征矢量和特征值可以用如下简化形式表示：

$$\boldsymbol{AX}=\boldsymbol{X\Lambda} \tag{7.50}$$

其中，矩阵 $\boldsymbol{X}$ 的列是各个特征矢量：

$$\boldsymbol{X}=[\boldsymbol{x}_1,\boldsymbol{x}_2,\boldsymbol{x}_3,\cdots,\boldsymbol{x}_n] \tag{7.51}$$

矩阵 $\boldsymbol{\Lambda}$ 是一个对角矩阵，其对角线上的值就是矩阵 $\boldsymbol{A}$ 的特征值：

$$\boldsymbol{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & 0 & \cdots & 0 \\ 0 & \lambda_2 & 0 & \cdots & 0 \\ 0 & 0 & \lambda_3 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & \lambda_n \end{bmatrix} \tag{7.52}$$

经过一系列矩阵运算，可以把式（7.47）和式（7.50）结合起来，将矩阵指数表示为

$$e^{At} = \boldsymbol{X} e^{\Lambda t} \boldsymbol{X}^{-1} \tag{7.53}$$

有关这个方程的完整推导过程请参阅 Constantinides 和 Mostoufi 的著作（1999）。

结合式（7.46）和式（7.53），就可以用特征值和特征矢量表示线性微分方程组的解，为

$$\boldsymbol{y} = [\boldsymbol{X} e^{\Lambda t} \boldsymbol{X}^{-1}] \boldsymbol{y}_0 \tag{7.54}$$

只要 $n \times n$ 矩阵 $\boldsymbol{A}$ 存在 n 个线性独立的特征矢量，就可以用这种方法求解。这个条件等价于矩阵 $\boldsymbol{X}$ 必须是非奇异的，其逆矩阵存在。可以利用附录 C 中建立的方法或者直接用下面介绍的 MATLAB 内置函数，计算矩阵 $\boldsymbol{A}$ 的特征值和特征矢量。

7.5.2 求解线性微分方程组的 MATLAB 函数

MATLAB 有如下用于计算矩阵指数以及特征值和特征矢量的函数：

expm1（A）：用 Pade 逼近算法（Burden 等人，1981）求 $\boldsymbol{A}$ 的矩阵指数。

expm2（A）：用泰勒级数求 $\boldsymbol{A}$ 的矩阵指数。实际使用时，这个方法通常较慢，也不准确。

expm3（A）：通过特征值和特征矢量求 $\boldsymbol{A}$ 的矩阵指数。这种方法的准确度由特征矢量矩阵的情况决定。

eig（A）：求矩阵 $\boldsymbol{A}$ 的特征值。

[X, LAMBDA] = *eig*（A）：生成形如式（7.52）的特征值的 LAMBDA 对角矩阵，并且生成形如式（7.51）的特征矢量矩阵 $\boldsymbol{X}$，以满足式（7.50）方程，也就是 $\boldsymbol{A}\ \boldsymbol{X}$ = X · LAMBDA。

利用这些 MATLAB 函数可以求解式（7.54）的方程组，程序如下：

```
syms t
A =[enter the elements of matrix A]
y0 =[enter the elements of vector y0]
[X,LAMBDA] =eig(A)
```

```
y = X * expm(LAMBDA * t) * X^ - 1 * y0
```

下面的例 7.3 进一步说明了这些 MATLAB 函数的用法。

例 7.3 药物吸收动力学

问题陈述：

体内药物吸收的机制可以用模型来仿真，如图 7.6 所示为最简单的一种模型，由 3 个过程组成。

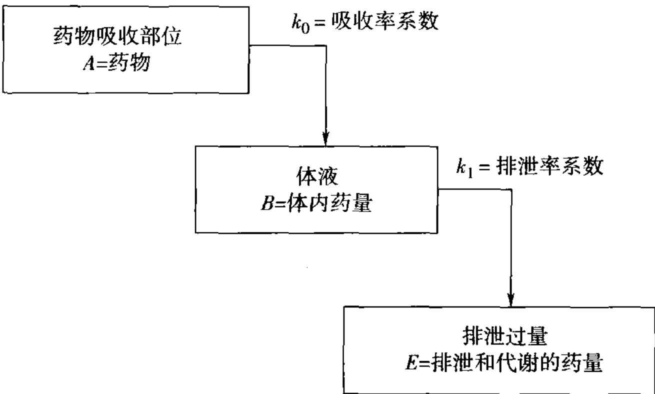


图 7.6 药物吸收和排泄的机制

假设所有体液作为一个单位看待。下面分别建立描述这 3 个过程的非稳态物质平衡的线性常微分方程。其中，描述吸收部位药物变化率的方程为

$$\frac{\mathrm{d}A}{\mathrm{d}t} = -k_0 A, \qquad A(0) = A_0 \tag{7.55}$$

描述体内药物变化率的方程为

$$\frac{\mathrm{d}B}{\mathrm{d}t} = k_0 A - k_1 B, \qquad B(0) = 0 \tag{7.56}$$

描述药物排泄变化率的方程为

$$\frac{\mathrm{d}E}{\mathrm{d}t} = k_1 B, \qquad E(0) = 0 \tag{7.57}$$

这 3 个方程组成了一阶线性常微分方程组，其解 $A(t)$、$B(t)$ 和 $E(t)$ 分别对应于给药浓度、体内药物浓度以及药物排泄浓度。如果已知 $k_0 = 0.01\mathrm{min}^{-1}$，$k_1 = 0.035\mathrm{min}^{-1}$。请分别用解析法和数值法求解该方程组，同时计算体内药物浓度达到最大值时的时间 t_{max}，以及体内药物达到的最大浓度 $B_{max} = B(t_{max})$，并作出这 3 个药物浓度随时间变化的曲线。

解：

(a) 微分方程组的解析解可以用如下 MATLAB 符号数学工具箱的指令

dsolve 来求取，即

```
>>[A,B,E]=dsolve('DA=-k0*A','DB=k0*A-k1*B','DE=k1*
B','A(0)=A0','B(0)=0','E(0)=0');
>>A=simplify(A)
A=
A0*exp(-k0*t)
>>B=simplify(B)
B=
k0*A0*(-exp(-k1*t)+exp(-k0*t))/(-k0+k1)
>>E=simplify(E)
E=
-A0*(exp(-k0*t)*k1-k1+k0-exp(-k1*t)*k0)/(-k0+k1)
```

由这些输出结果可知 A、B 和 E 的解析解为

$$A(t)=A_0e^{-k_0t}$$

$$B(t)=\frac{k_0A_0}{k_1-k_0}(e^{-k_0t}-e^{-k_1t})$$

$$E(t)=\frac{-A_0(k_1e^{-k_0t}-k_0e^{-k_1t})+A_0(k_1-k_0)}{(k_1-k_0)}$$

由质量守恒定律可知：

$$A(t)+B(t)+E(t)=A_0+B_0+E_0$$

用以下 MATLAB 指令很容易验证这个等式：（注意，本题中 B_0 和 E_0 都等于0）

```
>>simplify(A+B+E)
ans=
A0
```

求 $B(t)$ 的导数，并将其值设为0，代入 $k_0=0.01\text{min}^{-1}$ 和 $k_1=0.035\text{min}^{-1}$，可以解得 t_{max} 的值为

```
>>dB=diff(B)
dB=
k0*A0*(k1*exp(-k1*t)-k0*exp(-k0*t))/(-k0+k1)
>> tmax=solve(dB,'t')
tmax=
```

```
log(k1/k0)/(-k0+k1)
>> k0=0.01;k1=0.035;
>> eval(tmax)
ans =
    50.1105
```

此结果说明大约在用药之后50min时体内药物浓度达到最大值。

(b) 下面分别用式(7.54)的特征值-特征矢量算法以及式(7.46)的矩阵指数算法求本题的数值解。如下为MATLAB脚本，程序名是example7_3b.m。

```
% example7_3b.m-Solution of the drug absorption problem,
% both symbolically and numerically, using the eigenvalue-
% eigenvector method and the matrix exponential method.

clc; clear all;
syms c t
% Constants
k0=0.01; k1=0.035;
disp('Initial concentrations:')
c0=[1; 0; 0]
disp(' '); disp('Matrix of coefficients:')
K=[-k0 0 0; k0 -k1 0; 0 k1 0]

% Eigenvalue-eigenvector method
[X,lambda]=eig(K);
disp(' '), disp('Eigenvectors (each column of matrix X):'), X
disp(' ')
disp('Eigenvalues (on the diagonal of matrix lambda):'), lambda
disp(' '), disp('Inverse of X:'), X^-1
disp(' '); disp('Concentrations using eigenvalue-eigenvector meth-
od:')
c=X*expm(lambda*t)*X^-1*c0

% Evaluate concentration profiles
t=[0:100]; c=eval(c);
% Find the maximum concentration and time of drug in the body
```

```
[Cmax,tm] = max(c(2,:));
fprintf('\nMaximum concentration in the body = % 6.4f at tmax = %
4.2f min. \n',Cmax, tm - 1)
% Plot the results
clf; figure(1); h = plot(t,c(1,:), t,c(2,:),':',t,c(3,:),'--');
title('Eigenvalue-Eigenvector Solution')
ylabel('Concentration'); xlabel('Time, min'); legend('C_A','C_B
','C_C')

% Matrix exponential method
disp(' '); disp('Concentrations using matrix exponential method:')
syms t
c = expm(K * t) * c0
t = [0:100]; c = eval(c);

% Plot the results
figure(2); h = plot(t,c(1,:), t,c(2,:),':',t,c(3,:),'--');
title('Matrix Exponential Solution')
xlabel('Time, min'); ylabel('Concentration'); legend('C_A','C_B
','C_C')
```

输出结果（见图 7.7）：

```
Initial concentrations:
c0 =
    1
    0
    0

Matrix of coefficients:
K =
   -0.0100         0         0
    0.0100   -0.0350         0
         0    0.0350         0
```

```
Eigenvectors (each column of matrix X):
X =

         0         0    0.5661
         0    0.7071    0.2265
    1.0000   -0.7071   -0.7926

Eigenvalues (on the diagonal of matrix lambda):
lambda =
         0         0         0
         0   -0.0350         0
         0         0   -0.0100

Inverse of X:
ans =
    1.0000    1.0000    1.0000
   -0.5657    1.4142         0
    1.7664         0         0

Concentrations using eigenvalue-eigenvector method:
c =
                                      exp(-1/100*t)
   -2/5*exp(-7/200*t)+2/5*exp(-1/100*t)
  1+2/5*exp(-7/200*t)-7/5*exp(-1/100*t)
Maximum concentration in the body =0.1731 at tmax=50.00 min.
concentrations using matrix exponential method:
c =
                                      exp(-1/100*t)
   -2/5*exp(-7/200*t)+2/5*exp(-1/100*t)
  1+2/5*exp(-7/200*t)-7/5*exp(-1/100*t)
```

结果讨论:

不出所料，这两种方法的结果相同，并且都与解析法的结果一致。t_{max} 和 B_{max} 的值分别为 50min 和 0.1731。

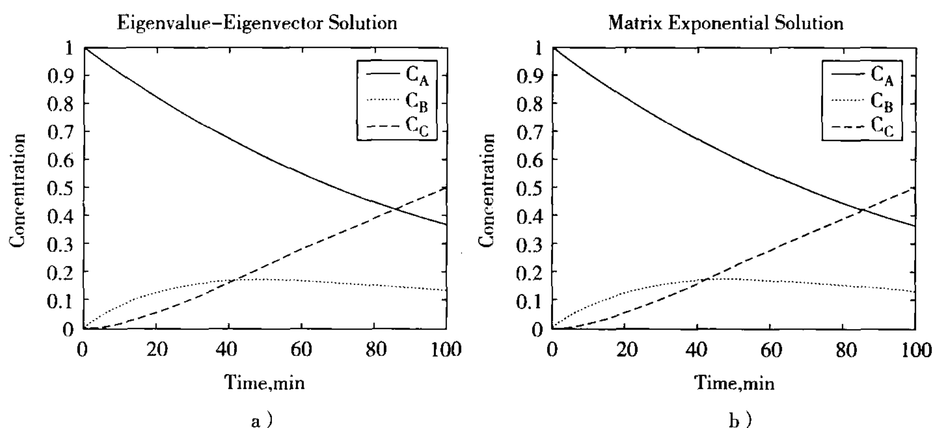


图 7.7　数值求解的输出结果

a）特征值-特征矢量求解法　b）矩阵指数求解法

7.6　稳态解及其稳定性分析

在求解微分方程组的数值解之前，我们先来考察方程组的稳态解。当变量随时间的变化率为 0 时，即系统进入了稳态。因此，将变量的时间导数设为 0，求解所得到的代数方程组，就可以得到微分方程组的稳态解。微分方程组很可能有多个稳态解，其中包括所有变量都等于 0 的平凡解。下面我们通过分析由两个非线性常微分方程组成的方程组来说明这些概念。方程如下：

$$\frac{\mathrm{d}N_1}{\mathrm{d}t}=f_1(N_1,N_2)$$
$$\frac{\mathrm{d}N_2}{\mathrm{d}t}=f_2(N_1,N_2) \tag{7.58}$$

稳态时，导数均为 0，因此：

$$f_1(N_1^*,N_2^*)=0,\quad f_2(N_1^*,N_2^*)=0 \tag{7.59}$$

式中　N_1^*，N_2^*——因变量的稳态值。

再定义两个稳态值的小偏差（即扰动）$\overline{N}_1$ 和 $\overline{N}_2$，使得

$$N_1=N_1^*+\overline{N}_1,\quad N_2=N_2^*+\overline{N}_2 \tag{7.60}$$

将式（7.60）代入式（7.58），可得

$$\frac{\mathrm{d}(N_1^*+\overline{N}_1)}{\mathrm{d}t}=f_1(N_1^*+\overline{N}_1,N_2^*+\overline{N}_2)$$

$$\frac{\mathrm{d}(N_2^* + \overline{N}_2)}{\mathrm{d}t} = f_2(N_1^* + \overline{N}_1, N_2^* + \overline{N}_2) \tag{7.61}$$

将方程左边分解成相应的两个导数，右边进行泰勒级数展开，则

$$\begin{aligned} \frac{\mathrm{d}N_1^*}{\mathrm{d}t} + \frac{\mathrm{d}\overline{N}_1}{\mathrm{d}t} &= f_1(N_1^*, N_2^*) + \left(\frac{\partial f_1}{\partial N_1}\right)^* \overline{N}_1 + \left(\frac{\partial f_1}{\partial N_2}\right)^* \overline{N}_2 + \text{高阶项} \\ \frac{\mathrm{d}N_2^*}{\mathrm{d}t} + \frac{\mathrm{d}\overline{N}_2}{\mathrm{d}t} &= f_2(N_1^*, N_2^*) + \left(\frac{\partial f_2}{\partial N_1}\right)^* \overline{N}_1 + \left(\frac{\partial f_2}{\partial N_2}\right)^* \overline{N}_2 + \text{高阶项} \end{aligned} \tag{7.62}$$

应用稳态条件（即稳态时，时间导数和f_1、f_2 两个函数均为 0），并且假设稳态值的扰动很小，因此可以忽略包含 $\overline{N}_1^2$，$\overline{N}_2^2$，$\overline{N}_1^3$，$\overline{N}_2^3$ 等的高阶项。于是，就可以将描述稳态值周围小扰动的方程组线性化，即式（7.62）简化为

$$\begin{aligned} \frac{\mathrm{d}\overline{N}_1}{\mathrm{d}t} &= \left(\frac{\partial f_1}{\partial N_1}\right)^* \overline{N}_1 + \left(\frac{\partial f_1}{\partial N_2}\right)^* \overline{N}_2 \\ \frac{\mathrm{d}\overline{N}_2}{\mathrm{d}t} &= \left(\frac{\partial f_2}{\partial N_1}\right)^* \overline{N}_1 + \left(\frac{\partial f_2}{\partial N_2}\right)^* \overline{N}_2 \end{aligned} \tag{7.63}$$

偏导数矩阵就是在稳态值邻近点计算的原始微分方程组的雅可比矩阵：

$$\boldsymbol{J}^* = \begin{bmatrix} \left(\frac{\partial f_1}{\partial N_1}\right)^* & \left(\frac{\partial f_1}{\partial N_2}\right)^* \\ \left(\frac{\partial f_2}{\partial N_1}\right)^* & \left(\frac{\partial f_2}{\partial N_2}\right)^* \end{bmatrix} \tag{7.64}$$

显然，式（7.63）是一个线性常微分方程组，可以写成如下形式：

$$\overline{\boldsymbol{N}}' = \boldsymbol{J}^* \overline{\boldsymbol{N}} \tag{7.65}$$

在 7.5 节已经证明，式（7.18）的线性常微分方程组可以用矩阵 $\boldsymbol{A}$ 的特征值求解。同理，式（7.65）的解取决于雅可比矩阵 $\boldsymbol{J}^*$ 的特征值。特征值可以是正实数、负实数，也可以是带正实部或负实部的复数。

特征值的一般形式为

$$\lambda_k = a_k \pm b_k \mathrm{i} \qquad k = 1,2,\cdots,n \tag{7.66}$$

式中 a_k——实部；

b_k——虚部的系数，$\mathrm{i} = \sqrt{-1}$。

记住，复数特征值总是以共轭复数对的形式出现。表 7.3 归纳了所有可能的特征值情况以及它们的稳定性分析，图 7.8 则显示了对应于各种情况的时间曲线以及 N_1 对于 N_2 的相图。负数特征值产生稳定解（第 1、2 种情况），而正数特征值引起不稳定（第 3、4 种情况）；复数特征值则产生解的振荡（第 2、4、6 种情况）。如果正实数特征值和负实数特征值都有，则解是亚稳定鞍点（第 5 种

情况）。最后，如果特征值为实部为 0 的复数，则解为中性稳定振荡（第 6 种情况）。

同样，可以分析含有 n（$n>2$）个独立变量的方程组的稳定性。这时，雅可比矩阵的大小为 $n \times n$，可以建立两两成对变量的相图。如果有必要，也可以建立三维相图。

表 7.3　基于雅可比矩阵特征值的稳定性分析

情况序号	a_k	b_k	稳定性分析
1	都是负数	0	稳定、无振荡
2	都是负数	非 0	稳定、振荡
3	都是正数	0	不稳定、无振荡
4	都是正数	非 0	不稳定、振荡
5	正数和负数	0	亚稳定、鞍点
6	0	非 0	中性稳定、振荡

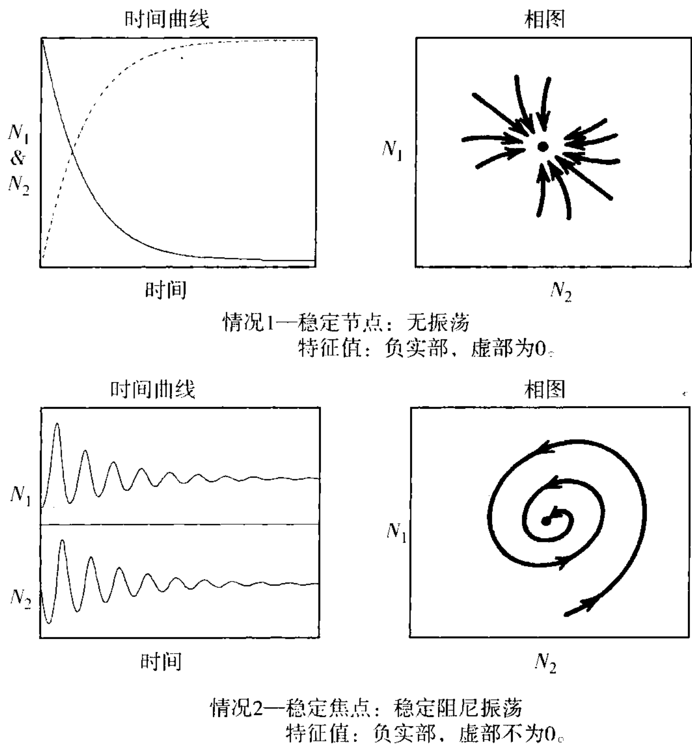


图 7.8　稳定性分析的时间曲线和相图

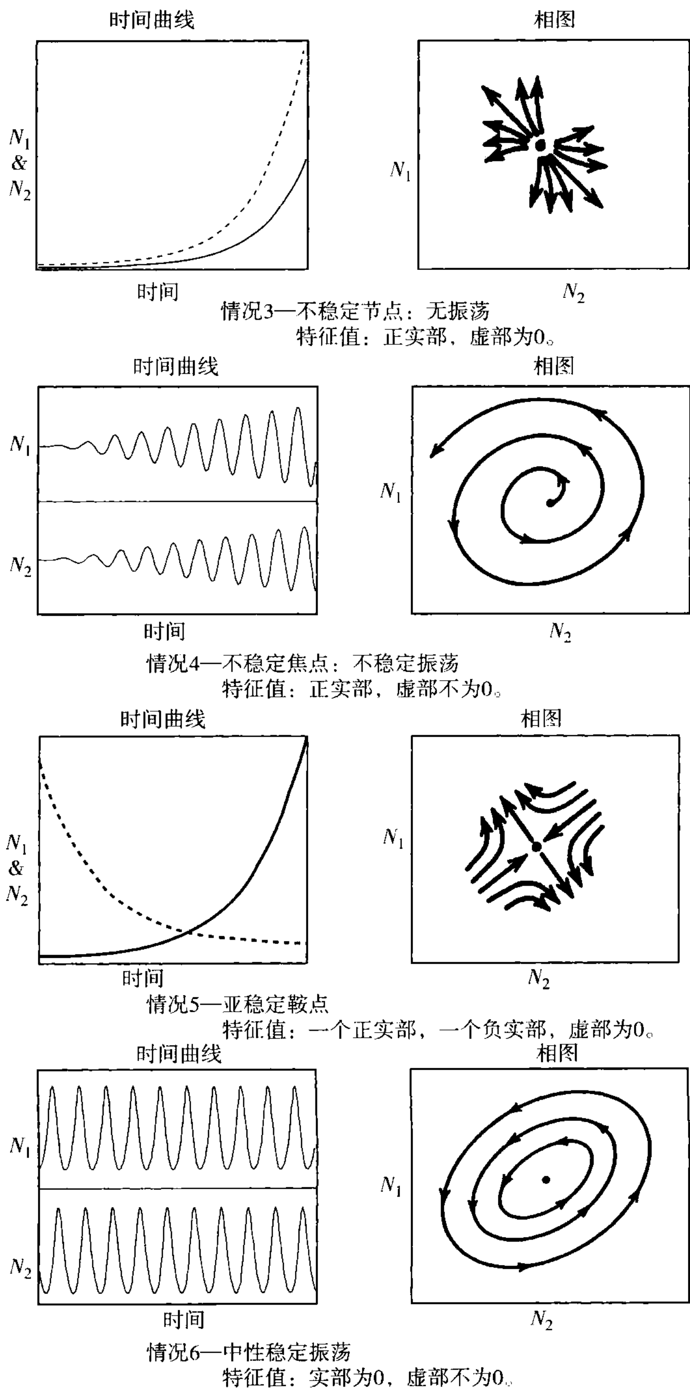


图 7.8 稳定性分析的时间曲线和相图（续）

7.7　数值稳定性和误差传播

有关微分方程数值积分的最重要的研究内容就是误差传播、稳定性以及解的收敛性。常微分方程求解有两类稳定性问题：固有稳定性和数值稳定性。固有稳定性由模型的数学公式决定，正如 7.6 节所述，该稳定性取决于微分方程雅可比矩阵的特征值。而数值稳定性决定于数值积分算法中的误差传播，误差传播则取决于求数值解的差分方程的特征根。本节将论述常微分方程数值积分的数值稳定性问题。

数值积分算法中存在着 3 种误差：截断误差、舍入误差和传播误差。截断误差的大小取决于近似解的无穷级数展开保留项的多少。增加级数的保留项或者减小积分步长 h，可以减小截断误差。常微分方程数值积分的求解方法越来越多，可以达到的计算准确度也越来越高，截断误差越来越小，但是这使得计算过程中所需执行的算术运算量迅速增加，也就伴随着舍入误差的累积越来越大。

第 3 章已经讲过，计算机只能用有限的有效位数存放数据，当计算机把数截成 n 位有效数字时，就引入了舍入误差。使用双精度数可以大大减小舍入误差。但是，即使是非常小的舍入误差也可能影响解的准确度，尤其是数值积分算法，需要向前或向后计算成千上万步，每一步计算都有舍入误差。

数值积分计算中的截断误差和舍入误差会积累和传播，就形成了传播误差。在某些情况下，传播误差会以指数形式增长，或者表现为振荡形式，导致计算所得的解严重偏离正确解。

图 7.9 显示了某个数值积分算法的传播误差。从一个已知的初始点 y_0 出发，该算法求得第一步的值 y_1，y_1 中包含了截断误差和舍入误差。当然，为了看得比较清楚，图中的误差放大了。第二步以 y_1 为起始点计算 y_2，因为 y_1 中已经含有截断误差和舍入误差，y_2 的值也就包含了传递过来的这部分误差，再加上第二步计算引入的新的截断误差和舍入误差。之后的每一步都有这样的过程。

数值积分算法中误差的传播有多个影响因素，是一个复杂的过程。舍入误差是传播误差的一部分，它完全由所使用的计算机的精度决定。截断误差则由选用的算法、积分步长的大小以及被积分函数的导数值决定。因此，必须结合所需求积的微分方程，分别考察每种算法的误差传播及其稳定性。适用于某种微分方程的方法可能并不能用于其他微分方程。

本书附录 E 对数值积分算法的稳定性问题进行了较为详细的论述，同时进一步讲述了其他问题，如步长控制、刚性微分方程组的求积等。

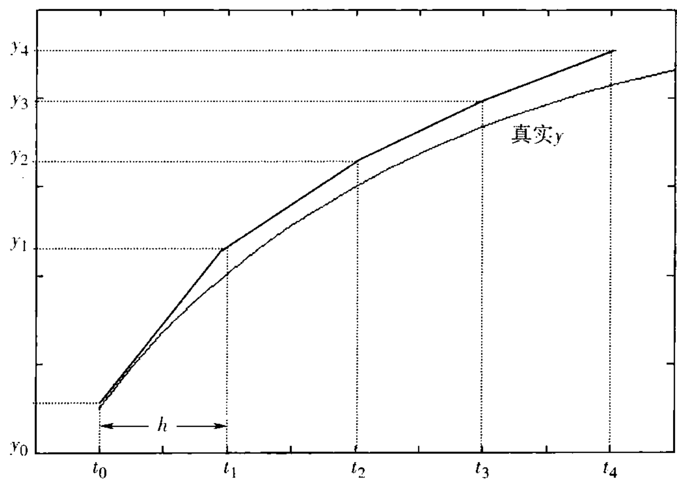


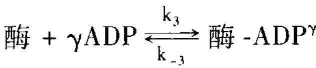
图 7.9 数值积分算法中的误差传播（为了能看得比较清楚，误差已被放大）

7.8 应用举例

例 7.4 代谢工程——活细胞糖酵解途径的建模仿真

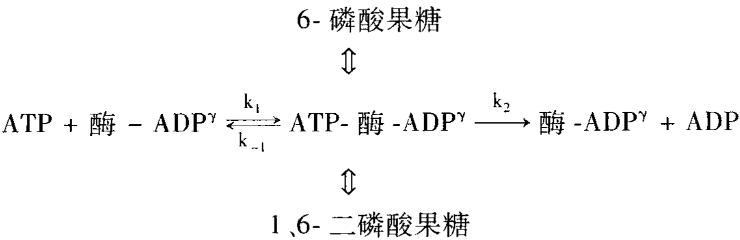
问题陈述：

糖酵解途径中很重要的一步就是6-磷酸果糖经磷酸化转变成1、6-二磷酸果糖。该反应由磷酸果糖激酶催化。磷酸果糖激酶是一种变构酶，可以被三磷酸腺苷（ATP）抑制，被二磷酸腺苷（ADP）或者腺苷酸（AMP）激活。当磷酸果糖激酶与 γ 分子 ADP 结合后，就被激活，其反应式为



$$\text{酶} + \gamma\text{ADP} \underset{k_{-3}}{\overset{k_3}{\rightleftharpoons}} \text{酶-ADP}^{\gamma}$$

已激活的活性复合酶再催化6-磷酸果糖转化为1、6-二磷酸果糖，并且将一分子 ATP 转变为一分子 ADP，即：



6-磷酸果糖

⇕

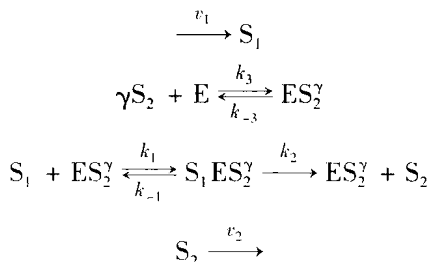
$$\text{ATP} + \text{酶} - \text{ADP}^{\gamma} \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} \text{ATP-酶-ADP}^{\gamma} \xrightarrow{k_2} \text{酶-ADP}^{\gamma} + \text{ADP}$$

⇕

1、6-二磷酸果糖

这就是 Keener 和 Sneyd 在1998年提出的 Sel'kov 模型。该反应的结果是产生更多的 ADP 分子进一步激活磷酸果糖激酶，因此，这是一个具有正反馈效应的反应。假设 ATP 的供应速率固定不变，为 v_1，并且，假设 ADP 离开反应的流速为 v_2，

则形成复合酶、消耗 ATP、并生成 ADP 的整个反应过程可以分为如下几步：



$$\xrightarrow{v_1} S_1$$

$$\gamma S_2 + E \underset{k_{-3}}{\overset{k_3}{\rightleftharpoons}} ES_2^{\gamma}$$

$$S_1 + ES_2^{\gamma} \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} S_1ES_2^{\gamma} \xrightarrow{k_2} ES_2^{\gamma} + S_2$$

$$S_2 \xrightarrow{v_2}$$

其中，S_1 代表 ATP 分子，S_2 代表 ADP 分子，E 代表磷酸果糖激酶。

Keener 和 Sneyd 应用质量作用定律，得到以下描述该反应动力学过程的常微分方程组：

$$\begin{aligned}
\frac{ds_1}{dt} &= v_1 - k_1 s_1 x_1 + k_{-1} x_2 \\
\frac{ds_2}{dt} &= k_2 x_2 - k_3 s_2^{\gamma} e + k_{-3} x_1 - v_2 s_2 \\
\frac{dx_1}{dt} &= -k_1 s_1 x_1 + (k_{-1} + k_2) x_2 + k_3 s_2^{\gamma} e - k_{-3} x_1 \\
\frac{dx_2}{dt} &= k_1 s_1 x_1 - (k_{-1} + k_2) x_2 \\
\frac{de}{dt} &= -\frac{dx_1}{dt} - \frac{dx_2}{dt}
\end{aligned} \tag{7.67}$$

其中，$s_1 = [S_1] = [ATP], s_2 = [S_2] = [ADP], e = [E], x_1 = [ES_2^{\gamma}], x_2 = [S_1ES_2^{\gamma}]$。方括号表示浓度。最后一个方程描述游离酶的变化率（de/dt），它来自于细胞内酶总量（e_0）的守恒方程，即假设酶的总量保持不变，为

$$e + x_1 + x_2 = e_0 \tag{7.68}$$

这些方程组成了一阶非线性常微分方程组。7.4 节介绍过这种方程组的求解方法，现在就用来求解这个糖酵解问题。

题目要求：

（a）由文献可知，糖酵解的速率是周期性振荡变化的。请利用以下初始条件和常数值，积分上述微分方程组，考察其计算结果是否真是如此。

初始条件：$s_1(0) = 1.0 \quad s_2(0) = 0.2 \quad x_1(0) = 0 \quad x_2(0) = 0 \quad e_0(0) = 1.4$

常量：$\gamma = 2.0 \quad v_1 = 0.003 \quad v_2 = 2.5{*}v_1 \quad k_1 = 0.1$

$k_{-1} = 0.2 \quad k_2 = 0.1 \quad k_3 = 0.2 \quad k_{-3} = 0.2$

注意，为了保持方程中所用单位的一致性，常数使用的时间单位为 s，浓度单位为 nM。

请作图显示5个独立变量的浓度曲线，分析其结果。并且，作出 s_1 和 s_2 的相图，分析相图的含义。

(b) 通过考察邻近稳态解的雅可比矩阵的特征值，对方程组进行稳定性分析。并说明特征值如何反映浓度随时间变化的振荡特性。

解：

(a) 方程组的求积。

下列程序 example7_4a.m，用 *ode*45 解法器计算微分方程组的积分，并作图显示其结果。

```
% example7_4a.m-Integration of the glycolysis model
% using the MATLAB function ode45.m to integrate the
% differential equations that are contained in the file:
% glycolysis_equations.m

clc; clear all;

% Set the initial conditions & time span
yzero=[1,.2,0,0,1.4];
tspan=[0 3000];

% Integrate the equations
[t,y]=ode45(@glycolysis_equations,tspan,yzero);
n=length(t);
% Plot concentration profiles
clf; figure(1); plot(t,y)
title('Concentration Profiles of Glycolysis')
xlabel('Time, s'); ylabel('Concentration')
text(530,1.35,'ATP (s_1)'); text(900,0.65,'ADP (s_2)')
text(1600,0.25,'Enzyme-ADP complex (x_1)')
text(1600,0.09,'ATP-Enzyme-ADP complex (x_2)')
text(1600,1.28,'free enzyme (e)')

% Plot phase diagrams
figure(2); plot(y(:,1),y(:,2))
title('Phase Plot of Glycolysis')
```

```
xlabel('ATP (s_1)'); ylabel('ADP (s_2)')
```

包含方程组的 MATLAB 函数（glycolysis_equations. m）

```
function dy=glycolysis_equations(t,y)
% glycolysis_equations.m
% Contains the glycolysis model for example7_4a

% Constants
gamma=2; neu1=0.003; neu2=2.5*neu1;
k1=0.1; k_1=2*k1; k2=0.1; k3=0.2; k_3=k3;
s1=y(1); s2=y(2); x1=y(3); x2=y(4); e=y(5);
% Equations
dy=[neu1-k1*s1*x1+k_1*x2
    k2*x2-k3*s2^gamma*e+k_3*x1-neu2*s2
    -k1*s1*x1+(k_1+k2)*x2+k3*s2^gamma*e-k_3*x1
    k1*s1*x1-(k_1+k2)*x2
    -(-k1*s1*x1+(k_1+k2)*x2+k3*s2^gamma*e-k_3*x1)-
(k1*s1*x1-(k_1+k2)*x2)];
```

积分结果如图 7. 10 所示

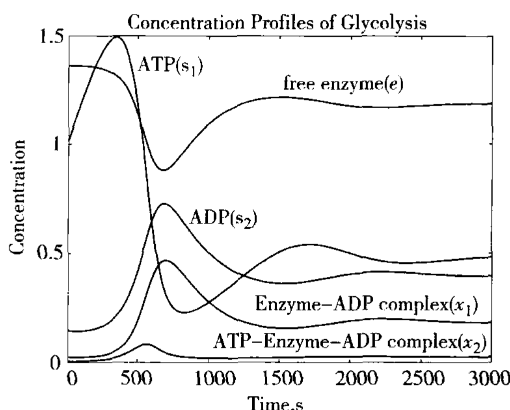


图 7. 10a　糖酵解反应的各个浓度曲线

由图 7. 10a 的糖酵解反应系统方程组的浓度曲线可见，在所给定的常量和初始条件下，系统开始时有一些振荡，但 3000s（即 50min）之后趋于稳态。图 7. 10b中 ADP 与 ATP 的相图显示了与图 7. 8 情况 2 类似的稳定焦点。

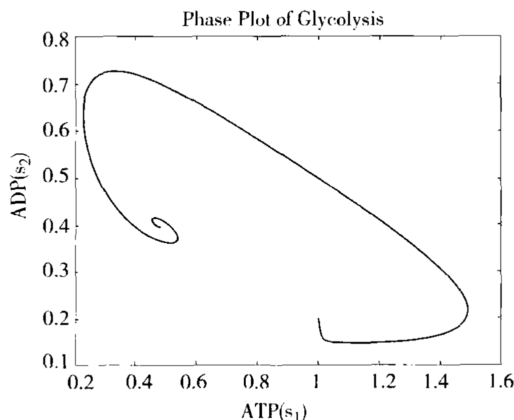


图 7.10b　糖酵解反应的相图

（b）糖酵解系统方程的稳态分析

下列程序 example7_4b.m 完成式（7.67）方程组的稳定性分析。首先，利用 MATLAB 指令 jacobian（dy，v）求取微分方程组的雅可比矩阵，其中，dy 为导数向量，v 为变量向量。然后，将导数设置为 0，并用 MATLAB 指令 *solve* 求解未知变量，以便计算微分方程的稳态解。最后，再用指令 eig 求得邻近稳态解的雅可比矩阵的特征值，用于考察稳态解的稳定性。

```
% example7_4b.m-Steady state analysis of the glycolysis model
% using MATLAB functions jacobian.m and eig.m

clc; clear all;

% Set the constants
e0 =1.4; gamma =2; neu1 =0.003; neu2 =2.5 * neu1;
k1 =0.1; k_1 =2 * k1; k2 =0.1; k3 =0.2; k_3 = k3;

% Evaluate the Jacobian matrix
syms s1 s2 x1 x2 e
disp('Steady State Analysis of the Glycolysis Equations:')
v =[s1, s2, x1, x2, e];
dy =[neu1 - k1 * s1 * x1 + k_1 * x2;
    k2 * x2 - k3 * s2^gamma * e + k_3 * x1 - neu2 * s2;
    - k1 * s1 * x1 + (k_1 + k2) * x2 + k3 * s2^gamma * e - k_3 * x1;
```

```
    k1 * s1 * x1 - (k_1 + k2) * x2;

 - ( - k1 * s1 * x1 + (k_1 + k2) * x2 + k3 * s2^gamma * e - k_3 * x1) - (k1 *
s1 * x1 - (k_1 + k2) * x2)];
J = jacobian(dy,v);
disp('The Jacobian matrix is:'), J
% Evaluate the steady state solution
[SteadyState] = solve('neu1 - k1 * s1 * x1 + k_1 * x2 = 0',...
    'k2 * x2 - k3 * s2^gamma * e + k_3 * x1 - neu2 * s2 = 0',...
    ' - k1 * s1 * x1 + (k_1 + k2) * x2 + k3 * s2^gamma * e - k_3 * x1 = 0',
...
    'k1 * s1 * x1 - (k_1 + k2) * x2 = 0',...
    'e + x1 + x2 = e0', 's1,s2,x1,x2,e');
disp(' '), disp('The steady state values of each variable are:')
disp('s1'),disp(SteadyState.s1),disp(' ')
disp('s2'),disp(SteadyState.s2),disp(' ')
disp('x1'),disp(SteadyState.x1),disp(' ')
disp('x2'),disp(SteadyState.x2),disp(' ')
disp('e '),disp(SteadyState.e), disp(' ')
n = length(SteadyState.s1);
disp('Value of each variable at the steady state(s):')
disp('         s1         s2         x1         x2         e')
for i = 1:n;
    s1 = eval(SteadyState.s1); s2 = eval(SteadyState.s2);
    x1 = eval(SteadyState.x1); x2 = eval(SteadyState.x2);
    e = eval(SteadyState.e);
    fprintf(' %2i  %9.4f  %9.4f  %9.4f  %9.4f  %9.4f  \n',...
    i, s1(i), s2(i), x1(i), x2(i), e);
end
for i = 1:n
    s1 = eval(SteadyState.s1); s2 = eval(SteadyState.s2);
    x1 = eval(SteadyState.x1); x2 = eval(SteadyState.x2);
    e = eval(SteadyState.e);
    fprintf('\nSteady state %2i \n',i)
    disp(' '); disp('Jacobian matrix at steady state:'), eval(J)
```

```
    disp(' '); disp('Eigenvalues of Jacobian at steady state:');
eig(eval(J))
end
```

稳态分析结果:

```
    Steady state Analysis of the Glycolysis Equations:
    The Jacobian matrix is:
    J =
    [ -1/10*x1,              0,      -1/10*s1,   1/5,          0]
    [        0, -2/5*s2*e-3/400,          1/5,  1/10, -1/5*s2^2]
    [ -1/10*x1,        2/5*s2*e, -1/10*s1-1/5,  3/10,  1/5*s2^2]
    [  1/10*x1,              0,       1/10*s1, -3/10,          0]
    [        0,       -2/5*s2*e,          1/5,     0, -1/5*s2^2]


    The steady satate values of each variable are:

    s1
    neu1*(k_1+k2)*(k3*exp(log(neu1/neu2)*gamma)+k_3)/k1/
exp(log(neu1/neu2)*gamma)/k3/(-neu1+e0*k2)

    s2
    neu1/neu2
    x1

    exp(log(neu1/neu2)*gamma)*k3*(-neu1+e0*k2)/k2/
    (k3*exp(log(neu1/neu2)*gamma)+k_3)

    x2
    neu1/k2

    e
    k_3*(-neu1+e0*k2)/k2/(k3*exp(log(neu1/neu2)*gamma)+k_3)
```

```
value of each variable at the steady state(s):
          s1          s2          x1          x2          e
1     0.4763      0.4000      0.1890      0.0300      1.1810

Steady state  1
Jacobian matrix at steady state:
ans =
 -0.0189         0     -0.0476    0.2000          0
       0   -0.1965      0.2000    0.1000    -0.0320
 -0.0189    0.1890     -0.2476    0.3000     0.0320
  0.0189         0      0.0476   -0.3000          0
       0   -0.1890      0.2000         0    -0.0320

Eigenvalues of Jacobian at steady state:
ans =
 -0.4859
 -0.3060
 -0.0015 +0.0044i
 -0.0015 -0.0044i
 -0.0000
```

对于给定的系统方程和常量，稳态分析结果显示：系统存在一个稳态解，达到稳态时，反应系统中各个主要成分的浓度如下：

[ATP] =0.4763　　[ADP] =0.4000

[酶-ADP 复合物] =0.1890

[ATP-酶-ADP 复合物] =0.0300

[游离酶] =1.1810

系统雅可比矩阵的特征值为：2 个负实数，2 个带负实数的复数和 1 个 0 特征值。这种特征值组合表明系统具有趋于稳态的阻尼振荡特性。0 特征值是酶总量质量守恒定律应用的结果。这些结论与浓度曲线反映的系统变化过程一致。

例 7.5 细胞膜和神经细胞动作电位的动力学过程

问题陈述：

Hodgkin 和 Huxley 在 20 世纪 40 年代和 50 年代所完成的工作（Hodgkin 和 Huxley，1952）获得了诺贝尔奖，他们的工作主要是研究钾、钠离子通道的开放和关闭，以及这些通道的活动在产生神经细胞动作电位中的作用。他们研究了枪

乌贼巨型轴突上外加电压对于 Na、K 离子通道的作用，并建立了描述离子通道动力学过程的数学模型。

有关 Hodgkin-Huxley 模型的文章和书非常多，其中 Keener 和 Sneyd（1998）的著作较简明地论述了这个模型。首先，他们把细胞膜模拟成电容和离子通道并联的形式，在没有外加电流的情况下，离子通道电流与电容电流之和必须为 0，因此，有方程：

$$C_m \frac{dV}{dt} + I_{ion} = 0 \tag{7.69}$$

式中 V——细胞膜内外的电压差；

C_m——细胞膜电容；

I_{ion}——离子通道电流。

在乌贼巨轴突以及很多神经细胞的细胞膜上，主要的离子电流为钠电流 I_{Na} 和钾电流 I_K，氯离子电流等其他电流合在一起称为漏电流 I_L。钠、钾离子通道的电流可以用如下电流-电压关系式计算：

$$I_{Na} = g_{Na}(V - V_{Na}) \tag{7.70}$$

$$I_K = g_K(V - V_K) \tag{7.71}$$

漏电流的计算式则为

$$I_L = g_L(V - V_L) \tag{7.72}$$

式中 g_{Na}，g_K——细胞膜钠离子和钾离子通道的电导；

g_L——漏电流的电导；

V_{Na}，V_K——钠和钾两种离子在细胞膜内外的浓度差引起的平衡电位；

V_L——氯离子等其他离子形成的漏电流为 0 时的膜电位。

钠和钾离子的平衡电位可以用能斯特（Nernst）方程计算：

$$V_{Na} = \frac{RT}{zF}\ln\left(\frac{[Na^+]_e}{[Na^+]_i}\right) \tag{7.73}$$

$$V_K = \frac{RT}{zF}\ln\left(\frac{[K^+]_e}{[K^+]_i}\right) \tag{7.74}$$

随着膜电压的变化，离子通道会开放或关闭，这种离子通道响应膜电压变化的特性是细胞电兴奋性的基础，对于神经生理学具有非常重要的意义。根据 Keener 和 Sneyd（1998）所述，流经一群离子通道的电流可以用以下方程描述：

$$I = \eta(V,t)\phi(V) \tag{7.75}$$

式中 $\eta(V,t)$ ——群体中开放通道所占的比例；

$\phi(V)$ ——单个通道的电流-电压（I-V）关系函数。

最简单的钾离子通道模型假设离子通道不是处于关闭状态（比例为 $1-\eta$），就是处于开放状态（比例为 η），即

$$\overbrace{(1-\eta)}^{\text{关闭}} \underset{\beta(V)}{\overset{\alpha(V)}{\rightleftharpoons}} \overbrace{(\eta)}^{\text{开放}} \tag{7.76}$$

于是，开放的离子通道的变化率就可以用以下微分方程描述：

$$\frac{\mathrm{d}\eta}{\mathrm{d}t} = \alpha(V)(1-\eta) - \beta(V)\eta \tag{7.77}$$

有时这个方程可以写成如下形式：

$$\tau_\eta(V)\frac{\mathrm{d}\eta}{\mathrm{d}t} = \eta_\infty(V) - \eta \tag{7.78}$$

式中　$\eta_\infty(V)$ —— η 的稳态值。

$\eta_\infty(V)$ 可以由式（7.77）求得，为

$$\eta_\infty(V) = \frac{\alpha}{\alpha+\beta} \tag{7.79}$$

τ_η 为趋向稳态的时间常数：

$$\tau_\eta = \frac{1}{\alpha+\beta} \tag{7.80}$$

Hodgkin 和 Huxley 在研究枪乌贼巨轴突时用了电压钳技术，就是在细胞膜上施加一个阶跃电压，把细胞膜电位从一个值固定到另一个值上，然后保持膜电位不变，同时测量外界所需提供的电流 I_{app}。根据测得的实验数据，为了拟合 S 形增加和指数式衰减的钾电导 g_K 的变化过程，Hodgkin 和 Huxley 提出了如下钾电导方程：

$$g_K = \bar{g}_K n^4 \tag{7.81}$$

同样，他们也提出了钠电导 g_{Na} 的方程，用于描述钠离子通道开放和关闭两个过程：

$$g_{Na} = \bar{g}_{Na} m^3 h \tag{7.82}$$

Keener 和 Sneyd（1998）对以上钾通道的机制进行了解释，他们把每个钾通道等价为包含 4 个“n”门的通道，只有当这 4 个门都打开时，钾离子才能流过通道，因此钾通道开放的概率为 n^4。他们也解释了钠通道的机制，把每个钠通道看成由 3 个“m”门和 1 个“h”门组成，每个门不是开就是关，也只有当这 4 个门都开放时，钠离子才能流过通道，因此钠通道开放的概率为 m^3h。

把式（7.69）~式（7.72）以及式（7.77）、式（7.81）和式（7.82）组合起来，就形成了如下完整的 Hodgkin-Huxley 模型：

$$C_m\frac{\mathrm{d}v}{\mathrm{d}t} = -\bar{g}_K n^4(v-v_K) - \bar{g}_{Na}m^3h(v-v_{Na}) - \bar{g}_L(v-v_L) + I_{app}$$

$$\frac{\mathrm{d}n}{\mathrm{d}t} = \alpha_n(1-n) - \beta_n n$$

$$\frac{dm}{dt} = \alpha_m(1-m) - \beta_m m$$
$$\frac{dh}{dt} = \alpha_h(1-h) - \beta_h h \tag{7.83}$$

其中膜电位 v 指的是偏离膜静息电位的值，即 $v = V - V_{eq}$，单位为 mV；电流密度 I 的单位为 $\mu A/cm^2$；各个离子通道电导的单位为 mS/cm^2；电容 C_m 的单位为 $\mu F/cm^2$。速率常数 α 和 β 的单位为 $(ms)^{-1}$，它们的方程是：

$$\alpha_n = 0.01\frac{10-v}{e^{\left(\frac{10-v}{10}\right)}-1} \qquad \beta_n = 0.125e^{\left(\frac{-v}{80}\right)}$$
$$\alpha_m = 0.1\frac{25-v}{e^{\left(\frac{25-v}{10}\right)}-1} \qquad \beta_m = 4e^{\left(\frac{-v}{18}\right)} \tag{7.84}$$
$$\alpha_h = 0.07e^{\left(\frac{-v}{20}\right)} \qquad \beta_h = \frac{1}{e^{\left(\frac{30-v}{10}\right)}+1}$$

各个门变量的稳态值和时间常数为

$$n_\infty = \frac{\alpha_n}{\alpha_n+\beta_n} \qquad m_\infty = \frac{\alpha_m}{\alpha_m+\beta_m} \qquad h_\infty = \frac{\alpha_h}{\alpha_h+\beta_h}$$
$$\tau_n = \frac{1}{\alpha_n+\beta_n} \qquad \tau_m = \frac{1}{\alpha_m+\beta_m} \qquad \tau_h = \frac{1}{\alpha_h+\beta_h} \tag{7.85}$$

该仿真模型的常量和初始条件如下：

$$\bar{g}_K = 36mS/cm^2 \qquad \bar{g}_{Na} = 120mS/cm^2 \qquad \bar{g}_L = 0.3mS/cm^2$$
$$v_K = -12mV \qquad v_{Na} = 115mV \qquad v_L = 10.6mV$$
$$v(0) = 8mV \qquad n(0) = 0.3177 \qquad m(0) = 0.0529$$
$$h(0) = 0.5961$$

其中，方程组（7.83）中 4 个变量（v，n，m 和 h）的初始条件根据 Hodgkin 和 Huxley 的陈述确定：他们指出“在动作电位产生过程的任意时刻，整条轴突上的膜电位都相等，轴突中没有轴向电流，因此，除了外加刺激期间存在净电流之外，膜的净电流总是为0。如果外加刺激只是一个 $t=0$ 时刻的短脉冲，则求解方程组（7.83）的动作电位波形时，I_{app} 应该为 0，并且有初始条件 $V=V_0$，n、m 和 h 的初始值则取 $t=0$ 时的静息稳态值。”

题目要求：

(a) 首先，核实初始条件 $n(0)$、$m(0)$ 和 $h(0)$ 的值，它们是 $v=0$ 时的静息稳态值。在 0 ~ 20ms 的时间范围内积分微分方程，设初始膜电位为 8mV。由于沿轴突方向没有电流，膜的净电流总是为 0，因此，电流密度为 0 $\mu A/cm^2$；膜电容设为 $1\mu F/cm^2$。请作出膜电位 v、门变量 n、m 和 h，以及离子通道电导

g_K 和 g_{Na}（方程（7.81）和（7.82））随时间变化的曲线。

(b) 计算 -100 ~ 100mV 膜电位范围内，各个时间常数和门变量(方程（7.85）)随膜电位变化的稳态值，并作出它们的曲线图。

解：

微分方程求积。

如下 example7 _5. m 程序首先利用方程组（7.85）计算门变量的初始值；然后用 MATLAB 函数 ode45. m 对包含在 hodgkin _ huxley _ equations. m 函数中的微分方程组求积分。同时，本程序也调用 rate _ constants. m 函数计算 α 和 β 的值，程序还计算各个时间常数和门变量的稳态值，并作图显示所有计算结果。

```
% example7_5.m-Simulation of the Hodgkin-Huxley model
% using MATLAB function ode45.m to integrate the differential
% equations that are contained in the file:
% hodgkin_huxley_equations.m


clc; clear all;
warning off MATLAB:divideByZero
% Evaluate the initial conditions for gating variables
v=0; [alpha_n,beta_n,alpha_m,beta_m,alpha_h,beta_h]=rate_
constants(v);
tau_n=1./(alpha_n+beta_n);
n_ss=alpha_n.*tau_n;
tau_m=1./(alpha_m+beta_m);
m_ss=alpha_m.*tau_m;
tau_h=1./(alpha_h+beta_h);
h_ss=alpha_h.*tau_h;
fprintf('\n The following initial conditions of the gating varia-
bles are used:')
fprintf('\n n_ss=%5.4g \n m_ss=%5.4g \n h_ss=%5.4g ', n_ss,
m_ss,h_ss)
fprintf('\n They are the resting steady state values of these varia-
bles (when v=0). ')
% Integrate the equations
yzero=[8,n_ss,m_ss,h_ss]; tspan=[0,20];
[t,y]=ode45(@hodgkin_huxley_equations,tspan,yzero);
```

```
% Evaluate the conductances
ggK=36; ggNa=120;
gK=ggK*y(:,2).^4; gNa=ggNa*y(:,3).^3.*y(:,4);

% Plot the results
clf; figure(1); plot(t,y(:,1),'k');
title('Time Profile of Membrane Potential in Nerve Cells')
xlabel('Time (ms)'); ylabel('Potential (mV)')
figure(2); plot(t,y(:,2:4));
title('Time Profiles of Gating Variables')
xlabel('Time (ms)'); ylabel('Gating variables')
text(7,0.6,'\leftarrow n(t)'); text(4.5,0.9,'\leftarrow m(t)');
text(7,0.25,'\leftarrow h(t)')
figure(3); plot(t,gK,t,gNa);
title('Time Profiles of Conductances')
xlabel('Time (ms)'); ylabel('Conductances')
text(7,6,'g_K'); text(3.6,25,'g_{Na}');

% Evaluate the rate constants
v=[-100:1:100];
[alpha_n,beta_n,alpha_m,beta_m,alpha_h,beta_h]=rate_con-
stants(v);

% Evaluating time constants and gating variables at steady state
tau_n=1./(alpha_n+beta_n);
n_ss=alpha_n.*tau_n;
tau_m=1./(alpha_m+beta_m);
m_ss=alpha_m.*tau_m;
tau_h=1./(alpha_h+beta_h);
h_ss=alpha_h.*tau_h;

% Plot the time constants
figure(4); plot(v,tau_n,v,tau_m,v,tau_h)
axis([-100 100 0 10])
title('Time Constants as Functions of Potential')
```

```
xlabel('Potential (mV)'); ylabel('Time constants (ms)')
text(-75,4,'\tau _n'); text(0,0.8,'\tau _m'); text(15,8,'\tau _h');

% Plot the gating variables at steady state
figure(5); plot(v,n_ss,v,m_ss,v,h_ss)
axis([-100 100 0 1])
title('Gating Variables at Steady State as Functions of Potential')
xlabel('Potential (mV)'); ylabel('Gating variables at steady state')
text(-35,0.1,'n _\infty'); text(25,0.4,'m _\infty');
text(-20,0.8,'h _\infty');
```

包含待求解方程组的 MATLAB 函数（hodgkin_huxley_equations.m）

```
function dy=hodgkin_huxley_equations(t,y)
% hodgkin_huxley_equations.m
% Contains the Hodgkin-Huxley model for example7_5

% Constants
ggK=36; ggNa=120; ggL=0.3;
vK= -12; vNa=115; vL=10.6;
Iapp=0; Cm=1;
% Equations
v=y(1); n=y(2); m=y(3); h=y(4);
[alpha_n,beta_n,alpha_m,beta_m,alpha_h,beta_h]=rate_con-
stants(v);

dy=[(-ggK*n^4*(v-vK)-ggNa*m^3*h*(v-vNa)-ggL*(v-vL)+
Iapp)/Cm
    alpha_n*(1-n)-beta_n*n
    alpha_m*(1-m)-beta_m*m
    alpha_h*(1-h)-beta_h*h];
```

计算速率常数的 MATLAB 函数（rate_constants.m）

```
function [alpha_n,beta_n,alpha_m,beta_m,alpha_h,beta_h]=
rate_constants(v)
```

```
% rate_constants.m
% Calculates the rate constants for the Hodgkin-Huxley model

alpha_n=0.01*(10-v)./(exp((10-v)/10)-1);
beta_n=0.125*exp(-v/80);
alpha_m=0.1*(25-v)./(exp((25-v)/10)-1);
beta_m=4*exp(-v/18);
alpha_h=0.07*exp(-v/20);
beta_h=1./(exp((30-v)/10)+1);
```

运行结果（曲线图见图 7.11）：

```
    The following initial conditions of the gating variables are
used:
    n_ss=0.3177
    m_ss=0.05293
    h_ss=0.5961
They are the resting steady state values of these variables (when v
=0).
```

结果讨论：

在 $t=0$ 时刻给细胞膜施加一个刺激电位，使细胞膜电位（即偏离膜静息电位的值）达到 8mV，超过了阈值，于是诱发产生了如图 7.11a 所示的自主动作电位过程，膜电位迅速上升并超过 100mV，然后下降回到静息电位，整个过程持续时间小于 20ms。该动作电位的产生机制如下：由于钠通道的“m”门具有很小的时间常数 τ_m（见图 7.11d），因此，$m(t)$ 的响应比较快，也就是钠通道的开放比较快，使得钠离子流入细胞，从而膜电位进一步上升。随着膜电位的上升，h_∞ 的值变为 0（见图 7.11e），于是，导致钠电导为 0（见图 7.11c），钠通道失活。但是，“h”门的时间常数 τ_h 较大，因此，它产生作用较慢。当膜电位上升时，电压门控的钾通道也开放，但是，与钠通道不同，它们的开放比较慢，钠通道关闭之后钾通道才完全开放。这以后钾通道保持开放状态，直到膜电位回复到接近其静息电位为止。

建议同学们做一下本章末尾的习题 7.1，并分析和解释所得到的结果。该习题外加了 $10\mu A/cm^2$ 的恒定电流。

例 7.6 干细胞分化动力学

问题陈述：

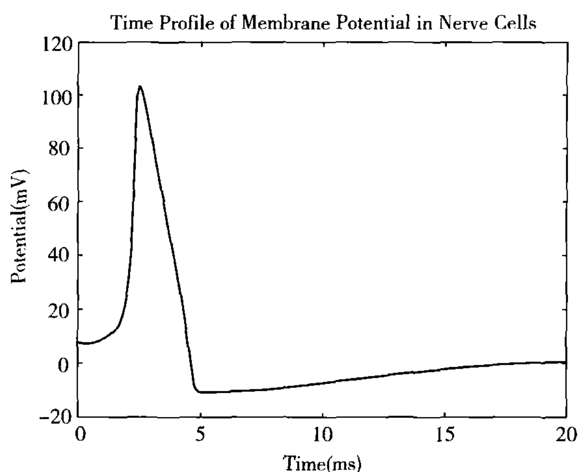


图 7.11a　神经细胞膜动作电位的时间曲线

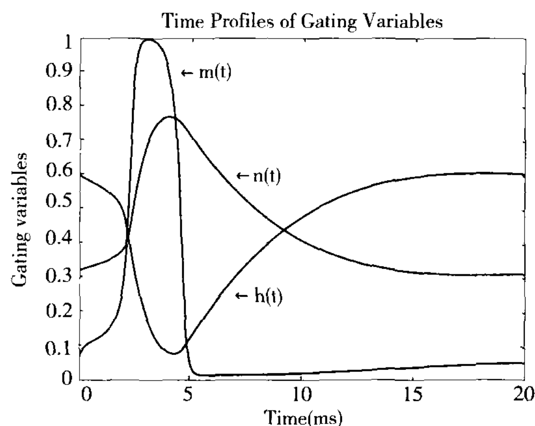


图 7.11b　各门变量的时间曲线

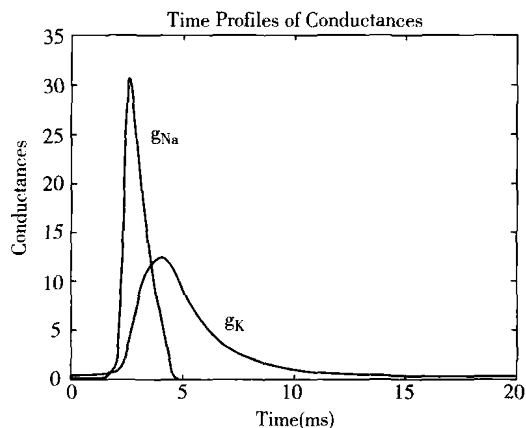


图 7.11c　离子通道电导的时间曲线

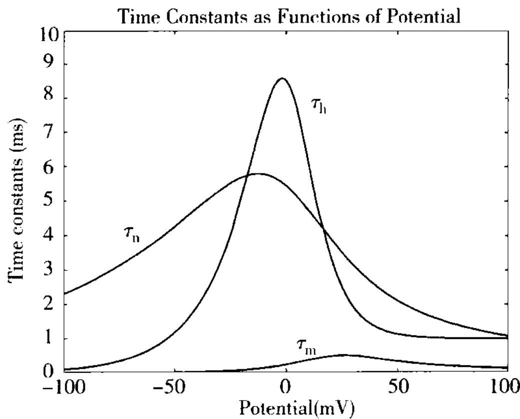


图 7.11d 时间常数随膜电位变化的曲线

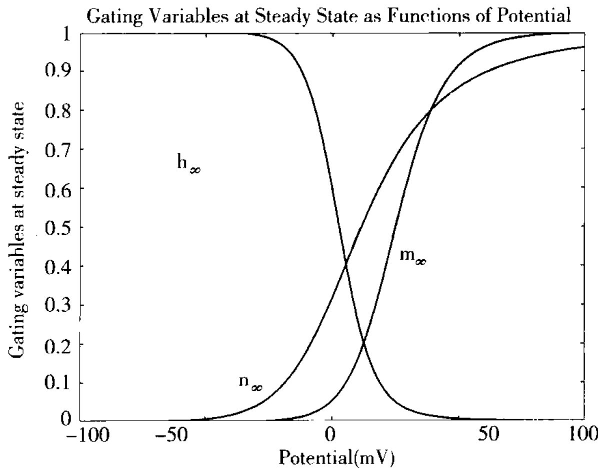


图 7.11e 门变量稳态值随膜电位变化的曲线

不断成长的胚胎体内的干细胞通过复制和分化，可以发育成为骨细胞、皮肤细胞、肝细胞、肌细胞等各种特异细胞。在成人体内，骨髓中含有干细胞，例如，产生红血球的造血干细胞、生产结缔组织细胞的间充质干细胞等。干细胞分化过程包括一系列细胞表型以及形态结构上的变化，尤其是在分化后期，这些变化更加明显，很容易直接辨别（Palsson 和 Bhatia，2004）。分化过程由任务递交开始，随后进行一系列有序的基因表达，每一次基因表达都使细胞的分化到达一个新的阶段。通过这一系列不断积累的变化，干细胞最终分化发育成完全成熟的特定细胞群。这些成熟细胞在体内完成它们的既定功能之后，最终要么凋亡，要么通过“转分化”的过程转变成其他类型的细胞。图 7.12 的框图描述了干细胞转变为完全成熟的特异细胞所经历的一系列过程。

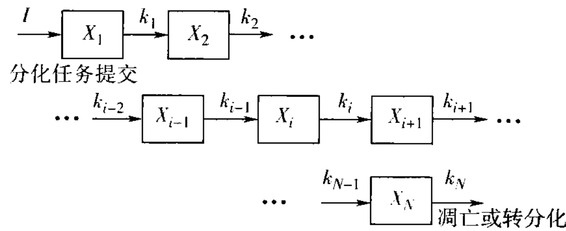


图 7.12　干细胞复制和分化的机制

其中，X_i 为处于第 i 个分化阶段的细胞数目（单位为细胞），I 为参加分化的细胞数目（单位为细胞/天），k_i 为细胞从第 i 个阶段进入第 $i+1$ 个阶段的转化率（单位为 1/天），N 为细胞分化需要经历的总的阶段数，可以多达 16 ~ 18 个阶段。

最后阶段（即第 N 个阶段）可以认为是细胞分化的既定目标，也就是细胞变成了成熟的特异细胞。如果最后这个阶段的转化率常数 k_N 为 0，则细胞无凋亡，也无转分化。

利用多房室模型很容易仿真细胞分化的动力学过程。假设每个阶段细胞内容物的分布都是均匀的，对模型的每个房室建立非稳态平衡方程，就可以得到以下常微分方程组：

$$\frac{dX_1}{dt} = I - k_1X_1$$

$$\frac{dX_2}{dt} = k_1X_1 - k_2X_2$$

$$\vdots$$

$$\frac{dX_i}{dt} = k_{i-1}X_{i-1} - k_iX_i$$

$$\vdots$$

$$\frac{dX_N}{dt} = k_{N-1}X_{N-1} - k_NX_N$$

该模型描述了干细胞从一个分化阶段到下一个分化阶段的转化过程，这里假设不存在干细胞的分裂和自我更新现象。本章最后所列习题中的 7.9 和 7.10 两个习题增加了干细胞的分裂和自我更新现象。

题目要求：

通过求解以上微分方程组的数值解，仿真如下干细胞分化过程：

（a）干细胞提交分化的速率为 $I=5000$ 细胞/天。假设分化过程有 10 个阶段，即 $N=10$。且最后阶段没有细胞损耗，即 $k_N=0$。求解微分方程组，并跟踪干细胞经历这 10 个分化阶段的过程。初始条件如下：

$I = 5000$ 细胞/天，

X_i (0) $=0$ 细胞 ($i=1, \cdots, N$)

$k_i = 2.2$ 天$^{-1}$ ($i=1, \cdots, N-1$)

$k_N = 0$,（无细胞死亡或转分化）

分析讨论各个时间曲线，并说明这个分化过程是否可以达到一个稳态。

（b）假设无新细胞加入分化过程，即 $I=0$，并且处于第一个分化阶段的初始细胞数目为 $X_i(0) = 5000$ 。设这些细胞经历与（a）相同的各个分化阶段，分化的最后阶段也无细胞损耗。请求解微分方程组，并跟踪这些细胞经历 10 个分化阶段的过程。初始条件如下：

$I = 0$ 细胞/天，

X_1 (0) $=5000$ 细胞，X_i (0) $=0$ ($i=2, \cdots, N$)

$k_i = 2.2$ 天$^{-1}$ ($i=1, \cdots, N-1$)

$k_N = 0$,（无细胞死亡或转分化）

分析讨论各个时间曲线，并说明完成整个分化过程需要多少天。

（c）除了第 10 个分化阶段存在细胞死亡之外，其他条件都与（a）相同。请分析各个时间曲线，并预测系统的稳态特性。初始条件如下：

$I = 5000$ 细胞/天，

X_i (0) $=0$ 细胞 ($i=1, \cdots, N$)

$k_i = 2.2$ 天$^{-1}$ ($i=1, \cdots, N$)

解：

以下是求解所有 3 种分化过程的 MATLAB 程序及函数：

```
% example7 _ 6. m-Solution of the stem cell differentiation model
% using MATLAB function ode45. m to integrate the differential
% equations that are contained in the file:
% cell _ differentiation _ equations. m


clc; clear all;
% Set the number of stages & time span
N =10; tzero =0; tmax =10; tspan =[tzero:0. 1:tmax];
% Case (a): With continuous input; no death
I =5000;              % Input
Xzero = zeros(N,1);            % Initial conditions
k =2.2 * ones (N - 1,1); k (N) = 0; % Transiton rate constants, no
death
```

```
% Integrate the equations
[t,X] =ode45('cell _ differentiation _ equations',tspan,Xzero,[ ],
N,I,k);
% Pseudo steady state values for stages 1 to N-1
SS=I/k(1); X _ last=X(length(X),N);
disp('Case (a)')
fprintf('The pseudo steady state number of cells in stages % 1d to%
2d=% 4.0f',1,N-1,SS)
fprintf('\nThe number of cells in stage % 2d, at % 2d days=% 4.0f \
n',N,tmax,X _ last)
% Plot concentration profiles
clf; figure(1); subplot(2,1,1), plot(t,X(:,1:1:N-1))
title(['Continuous input (I=',num2str(I),…
    '); no death (k(1:', num2str(N-1),')=',num2str(k(1)),…
    ', k(', num2str(N),')=',num2str(k(N)),') '])
text(0.4,SS,'i=1'); text(0.45*tmax,SS/2,['i=', num2str(N-1)]);
xlabel('Time, days'); ylabel('Number of cells');
subplot(2,1,2), plot(t,X(:,N)/1000)
axis([tzero, tmax, 0, 1.1*X _ last/1000])
text(tmax/2,X _ last/2000,['i=', num2str(N)]);
xlabel('Time, days'); ylabel('Number of cells (thousands)');
% Case (b): With no new input; no death
I=0;                          % Input
Xzero=zeros(N,1); Xzero(1)=5000; % Initial conditions
% Integrate the equations
[t,X]=ode45('cell _ differentiation _ equations',tspan,Xzero,[ ],
N,I,k);
% Steady state values for stages 1 to N-1
SS=I/k(1);
X _ last=X(length(X),N);
disp('Case (b)')
fprintf('The steady state number of cells in stages % 1d to % 2d=%
4.0f',1,N-1,SS)
fprintf('\nThe final number of cells in stage % 2d at % 2d days=%
4.0f \n',N,tmax,X _ last)
```

```
% Plot concentration profiles
figure(2); plot(t,X(:,1:1:N))
title(['No new input (I = ', num2str(I),...
    '); no death (k(1:', num2str(N-1),') = ', num2str(k(1)),...
    ', k(', num2str(N),') = ', num2str(k(N)),')'])
text(0.4,0.8*X(1),'i=1');
text(0.45*tmax,X(1)/2,['i=',num2str(N)]);
xlabel('Time, days'); ylabel('Number of cells');
% Case (c): With continuous input; with death (or transdiff.)
I=5000;                    % Input
Xzero=zeros(N,1);               % Initial conditions
k(N)=k(1); % reset the death rate constant
% Transiton rate constants, with death
% Integrate the equations
[t,X]=ode45('cell_differentiation_equations',tspan,Xzero,[],
N,I,k);
% Pseudo steady state values for stages 1 to N-1
SS=I/k(1);
X_last=X(length(X),N);
disp('Case (c)')
fprintf('The steady state number of cells in all stages =% 4.0f',SS)
% Plot concentration profiles
figure(3); plot(t,X(:,1:1:N))
axis([tzero, tmax, 0, 1.1*I/k(1)])
title(['Continuous input (I = ',...
    num2str(I), '); with death (k(1:', num2str(N),...
    ') =',num2str(k(1)),')'])
text(0.4,SS,'i=1'); text(0.5*tmax,SS/2,['i=', num2str(N)]);
xlabel('Time, days'); ylabel('Number of cells');
```

包含微分方程组的 MATLAB 函数（cell_differentiation_equations.m）

```
function dX=cell_differentiation_equations(t,X,flag,N,I,k)
% cell_differentiation_equations.m
% Contains the equations for example7_6
```

```
% Equations
dX(1) = I - k(1) * X(1);
for i = 2:N
    dX(i) = k(i-1) * X(i-1) - k(i) * X(i);
end
% Convert to column vector
dX = dX';
```

分化过程（a）的结果和讨论：

输出结果为：

Case（a）

The pseudo steady state number of cells in stages 1 to 9 = 2273

The number of cells in stage 10, at 10 days = 29547

图 7.13 所示是这个分化过程的仿真结果。图中上半部分是第 1 到第 9 阶段的时间曲线。第 1 阶段中恒定不变的干细胞输入（即 $I = 5000$ 细胞/天）使得细胞分化的前 9 个阶段在 10 天之内就达到稳态，稳态时每个阶段具有的细胞数为

$$X_i^* = \frac{I}{k_i} = \frac{5000\ 细胞/天}{2.2/天} = 2273\ 细胞$$

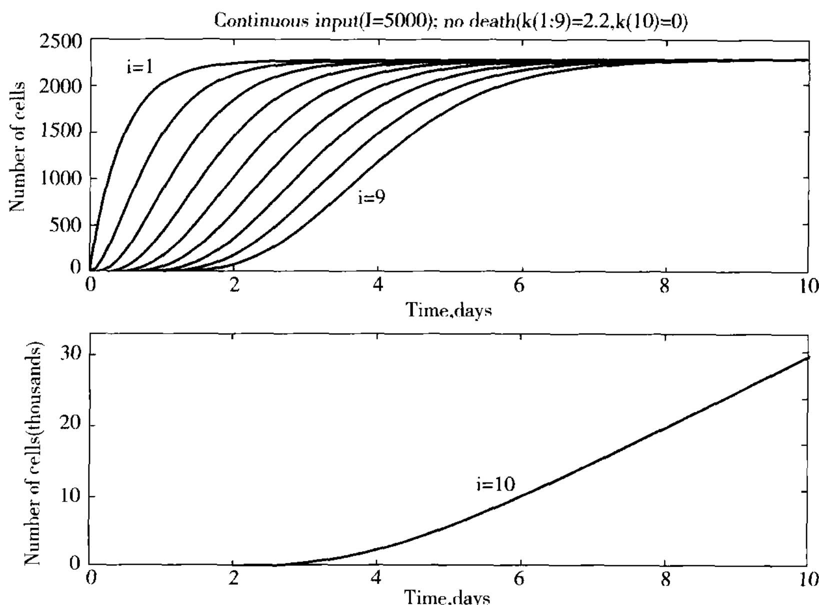


图 7.13　在输入恒定并且最后阶段无细胞损耗的情况下，
各个分化阶段的细胞数目随时间变化的曲线

把前 9 个微分方程的导数设为 0（即稳态），求解稳态时的细胞数 X_i^*，就可以得到这个数值。但是，由于没有细胞损耗（$k_{10}=0$），最后第 10 阶段的微分方程不存在稳态。如果设 $\frac{dX_{10}}{dt}=0$，就得到 $I=0$，显然是错误的。因此，我们称这个分化过程具有“准稳态”。如图 7.13 的下半部分所示，第 10 天时，处于最后阶段的细胞数目为 29547，并且还在继续增加。这个最后阶段是细胞分化过程的既定目标，因此，细胞就是应该在这个阶段不断产生积累。

分化过程（b）的结果和讨论：

输出结果为：

```
Case (b)
The steady state number of cells in stages 1 to 9 =0
The final number of cells in stage 10 at 10 days =4997
```

图 7.14 是该分化过程的仿真结果。由于没有新加入的干细胞，分化的最后阶段也没有细胞损耗，因此，如图 7.14 所示，现有的细胞就一个接一个地完成每个分化阶段，最后，都聚集到最后阶段。由于 $I=0$，所以，第 1 到第 9 阶段的稳态细胞数目均为 0，即

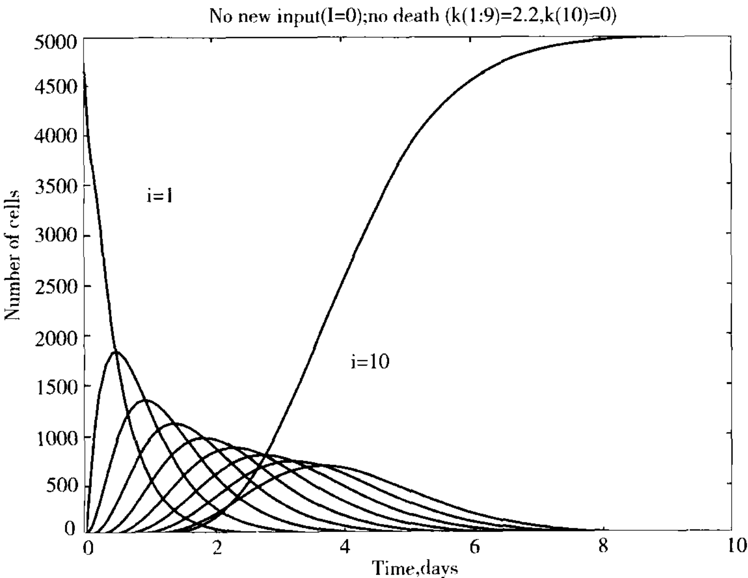


图 7.14　在既无新细胞输入又无细胞损耗的情况下，各个分化阶段的细胞数目随时间变化的曲线

$$X_i^* = \frac{I}{k_i} = \frac{0\text{ 细胞}/\text{天}}{2.2/\text{天}} = 0\text{ 细胞}$$

既然初始细胞数为 5000，在分化过程中又无任何细胞损耗，因此，第 10 阶段的最终细胞数约为 5000。

分化过程（c）的结果和讨论：

输出结果为：

```
Case (c)
The steady state number of cells in all stages =2273
```

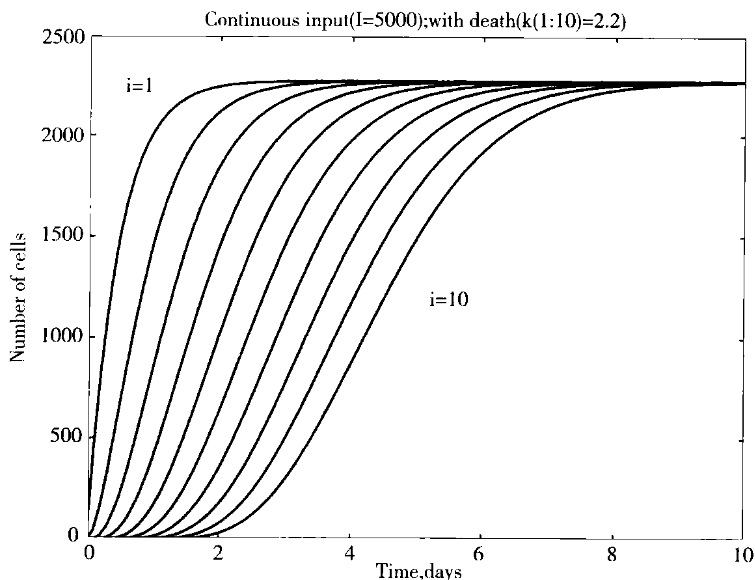


图 7.15　在输入恒定并且存在细胞损耗的情况下，各个分化阶段的细胞数目随时间变化的曲线

图 7.15 是这个分化过程的仿真结果。该细胞分化过程中，新干细胞提交分化的速率恒定不变，为 $I = 5000$ 细胞/天。并且，第 10 阶段也有细胞损耗。这样，所有 10 个阶段的稳态细胞数均为

$$X_i^* = \frac{I}{K_i} = \frac{5000\text{ 细胞}/\text{天}}{2.2/\text{天}} = 2273\text{ 细胞}$$

细胞不断地从一个阶段分化到下一个阶段，并且在分化到最后阶段之后死亡。理论上，这样的细胞分化过程会在生物体中持续一生。

例 7.7　组织工程——表皮细胞迁移模型

简介：

组织工程的研究方向之一就是设计并制造合适的多孔基质（即多孔膜），仿真表皮的特性，作为康复治疗的支架，用于促进真皮的再生，从而提高创伤和烧伤皮肤的愈合速度。要使细胞向创伤面集中，进入所植入的支架中并促使组织再生成功，就必须有细胞迁移。据了解，细胞迁移与特定的细胞表面受体和胞外基质上的细胞内吞配体之间的相互作用有关，皮肤表皮细胞（即角化细胞）与微载体中存在的配体之间形成配体-受体复合物，可以启动和促进胞吞作用，也就是细胞吸收基质分子，从而大幅度提高细胞的运动。

Tjia 和 Moghe 两人以动力学原理为基础，利用与常规 Michaelis-Menten 动力学方程相似的扩散反应方程，建立了细胞与配体相互作用以及胞吞耦合的动力学模型（Tjia 和 Moghe，2002c）。

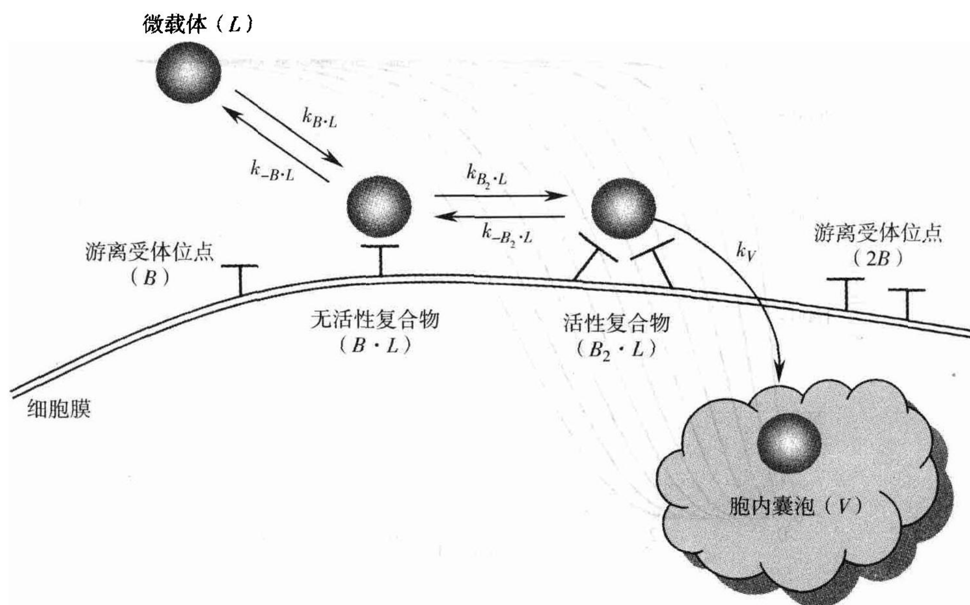


图 7.16　细胞与配体的相互作用

模型的机制：

图 7.16 显示了细胞与配体相互作用的机理，其中的每一步过程描述如下：

1）含有配体的微载体（L）作用于细胞表面的游离受体位点（B），形成无活性复合物（$B\cdot L$），即



$$L+B\underset{k_{-B\cdot L}}{\overset{k_{B\cdot L}}{\rightleftharpoons}}B\cdot L \tag{7.86}$$

2）无活性复合物再与第二个受体结合，形成活性复合物（$B_2\cdot L$），即



$$B\cdot L+B\underset{k_{-B_2\cdot L}}{\overset{k_{B_2\cdot L}}{\rightleftharpoons}}B_2\cdot L \tag{7.87}$$

3）活性复合物被细胞吸收，变成胞内囊泡。复合物被吸收之后，微载体也就脱离了细胞膜受体，使受体回收，重新成为细胞膜表面的游离受体。基于这个机制，胞吞吸收的速率和受体回收的速率合在一起，用参数 k_V 表示，即



$$B_2 \cdot L \xrightarrow{k_V} V + 2B \tag{7.88}$$

数学模型的建立：

细胞产生迁移之后，可以获得新的微载体用于胞吞作用，因此，细胞迁移会使细胞接触微载体的机率发生变化。与半无限平面中分子扩散的机理类似，可以导出如下细胞迁移（Migration）的速率方程：

$$\left.\frac{\mathrm{d}[L]}{\mathrm{d}t}\right|_{\text{Migration}} = \frac{\mu L_0}{A_{\text{cell}}} \tag{7.89}$$

式中　L——细胞遇到的有效配体的浓度；

μ——随机运动系数；

A_{cell}——细胞膜表面积；

L_0——微载体总浓度。

下面是包含细胞迁移和配体—受体结合这两个部分的整体模型：

1）细胞遇到的局部胞外微载体浓度［L］的变化由如下速率方程决定：

$$\frac{\mathrm{d}[L]}{\mathrm{d}t} = -k_{B\cdot L}[L][B] + k_{-B\cdot L}[B \cdot L] + \frac{\mu L_0}{A_{\text{cell}}} \tag{7.90}$$

该方程的第一项对应于反应式（7.86）的正向反应速率，第二项对应于反向反应速率，第三项则反应了方程（7.89）给出的细胞迁移速率。

2）由无活性微载体—受体复合物浓度［$B \cdot L$］的平衡，可以导出如下速率方程：

$$\frac{\mathrm{d}[B \cdot L]}{\mathrm{d}t} = k_{B\cdot L}[L][B] - k_{-B\cdot L}[B \cdot L] - k_{B_2\cdot L}[B \cdot L][B] + k_{-B_2\cdot L}[B_2 \cdot L] \tag{7.91}$$

3）活性微载体—受体复合物浓度［$B_2 \cdot L$］的变化速率为

$$\frac{\mathrm{d}[B_2 \cdot L]}{\mathrm{d}t} = k_{B_2\cdot L}[B \cdot L][B] - k_{-B_2\cdot L}[B_2 \cdot L] - k_V[B_2 \cdot L] \tag{7.92}$$

4）胞吞微载体的浓度［V］的变化速率为

$$\frac{\mathrm{d}[V]}{\mathrm{d}t} = k_V[B_2 \cdot L] \tag{7.93}$$

5）细胞膜上受体位点的总数［B_T］定为常数，即

$$[B_T] = [B] + [B \cdot L] + [B_2 \cdot L] \tag{7.94}$$

通过测量细胞对于覆盖了可吸收微载体的表面的实际清除（clearance）速

率，就可以了解细胞迁移的净效果。对于给定的初始表面颗粒浓度，细胞的表面清除速率等于两种复合物的产生速率与胞吞速率之和除以初始颗粒浓度，即：

$$\frac{\mathrm{d}[\text{clearance}]}{\mathrm{d}t}=\frac{1}{L_0}\left(\frac{\mathrm{d}[B\cdot L]}{\mathrm{d}t}+\frac{\mathrm{d}[B_2\cdot L]}{\mathrm{d}t}+\frac{\mathrm{d}[V]}{\mathrm{d}t}\right) \tag{7.95}$$

应该注意，模型中浓度项的单位是每单位细胞表面积上的颗粒数。另外，方程中［clearance］项的单位为（min）$^{-1}$。

为了简化模型，制定如下假设：

（a）无活性微载体—受体复合物的分解速率常数 $k_{-B\cdot L}$与结合速率常数 $k_{B\cdot L}$相比非常小，可以忽略不计。这实际上就是使反应式（7.86）不可逆。

（b）活性复合物［$B_2\cdot L$］一旦形成，其活性很高，立刻会被胞吞，因此，该复合物具有的浓度只能很小，可以认为是处于准稳态。也就是方程（7.92）的变化率可以假设等于0，于是，就可以得到［$B_2\cdot L$］：

$$[B_2L]=\frac{[B\cdot L][B]}{K_m} \tag{7.96}$$

式中　K_m——Michaelis-Menten 方程的分解常数。

定义为

$$K_m=\frac{k_{-B_2\cdot L}+k_V}{k_{B_2\cdot L}} \tag{7.97}$$

将这两个假设用于式（7.90）～式（7.95）这几个方程，消去［$B_2\cdot L$］[㊀]和［B］，就得到如下简化之后的模型方程：

$$\frac{\mathrm{d}[L]}{\mathrm{d}t}=-k_{B\cdot L}[L]([B_\mathrm{T}]-[B\cdot L])+\frac{\mu L_0}{A_\mathrm{cell}}$$

$$\frac{\mathrm{d}[B\cdot L]}{\mathrm{d}t}=k_{B\cdot L}[L]([B_\mathrm{T}]-[B\cdot L])-k_V\left(\frac{([B_\mathrm{T}]-[B\cdot L])[B\cdot L]}{K_m+2[B\cdot L]}\right)$$

$$\frac{\mathrm{d}[V]}{\mathrm{d}t}=k_V\left(\frac{([B_\mathrm{T}]-[B\cdot L])[B\cdot L]}{K_m+2[B\cdot L]}\right)$$

$$\frac{\mathrm{d}[\text{clearance}]}{\mathrm{d}t}=\frac{1}{L_0}\left(\frac{\mathrm{d}[B\cdot L]}{\mathrm{d}t}+\frac{\mathrm{d}[V]}{\mathrm{d}t}\right) \tag{7.98}$$

这个方程组定义了配体微载体促使细胞迁移增强的动力学数学模型，是一个常微分方程组，可以求解。下面是题目要求：

（a）基于 Tjia 和 Moghe 两人的实验结果（2002c），求 300min 时间段内各种物质的变化曲线并作图，分析所得到的结果。初始条件如下：

㊀ 原文此处为［$B\cdot L$］。—译者注

$$L_0 = 1.0\ \text{颗粒}/\mu m^2 \qquad [B\cdot L]|_0 = 0$$

$$[\text{clearance}]|_0 = 0 \qquad [V]|_0 = 0$$

常数为

$$B_T = 3.74\ \text{颗粒}/\mu m^2 \qquad \mu = 10\mu m^2/\min \qquad A_{cell} = 3400\mu m^2$$

$$K_m = 0.73\ \text{颗粒}/\mu m^2 \qquad k_V = 1.3\times 10^{-3}\min^{-1}$$

$$k_{B\cdot L} = 2.0\times 10^{-3}\mu m^2/(\text{颗粒}.\min)$$

（b）“采样率”（即细胞与配体的净结合率）定义为

$$(\text{采样率}) = \frac{d[B\cdot L]}{dt} - \frac{d[V]}{dt}$$

请说明这个采样率对于清除率 d ［clearance］/dt 的影响。

解：

以下是问题求解的程序 example7 _ 7. m 以及函数 cell _ migration _ equations. m。

```
% example7_7.m-Solution of the epidermal cell migration
% model using MATLAB function ode45.m to integrate the
% differential equations that are contained in the file:
% cell_migration_equations.m

clc; clear all;
% Set the time span
tspan=[0:1:300];
% Set the constants
BT=3.74; mu=10; A_cell=3400;
Km=0.73; kV=1.3e-3; kBL=2.0e-3;
% Set the initial conditions
yzero=[1, 0, 0, 0];
L0=yzero(1);
% Integrate the equations
[t,y]=ode45('cell_migration_equations',tspan,yzero,[],…
    BT,mu,A_cell,Km,kV,kBL,L0);

% Plot concentration profiles
figure(1); plot(t,y(:,1),'-',t,y(:,2),':',t,y(:,3),'-.',…
    t,y(:,4),'--')
```

```
title('Time profiles of epidermal cell migration')
xlabel('Time, min'); ylabel('Densities, number/\mum^2');
legend('L','B|\bf\cdot|L','V','clearance',2)
n = length(y);

% Evaluate the derivatives
for i = 1:n
dy(:,i) = feval('cell_migration_equations',t(i),y(i,:),flag,…
    BT,mu,A_cell,Km,kV,kBL,L0);
end
dy = dy';
rate_BL = dy(:,2);
rate_V = dy(:,3);
clearance_rate = dy(:,4);
sampling_rate = rate_BL-rate_V;

% Show the effect of microcarrier sampling rate on internalization
% and clearance rates
figure(2);
plot(sampling_rate * 1e3,clearance_rate * 1e3)
title('The effect of microcarrier sampling rate on clearance rate')
ylabel('Clearance rate, d[clearance]/dt x 10^3')
xlabel('Sampling rate, (d[B|\bf \cdot|L]/dt-d[V]/dt) x 10^3')
```

包含微分方程组的 MATLAB 函数（cell_migration_equations. m）

```
function dy = cell_migration_equations(t,y,flag,BT,mu,A_cell,
Km,kV,kBL,L0)
% cell_migration_equations.m
% Contains the equations for example7_7

% Equations
dy = [ - kBL * y(1) * (BT - y(2)) + mu * L0/A_cell
    kBL * y(1) * (BT - y(2)) - kV * ((BT - y(2)) * y(2))/(Km + 2 * y(2))
    kV * ((BT - y(2)) * y(2))/(Km + 2 * y(2))
    (kBL * y(1) * (BT - y(2)) - kV * ((BT - y(2)) * y(2))/(Km + 2 * y(2))…
```

```
+ kV * ((BT - y(2)) * y(2))/(Km + 2 * y(2)))/L0];
```

输出结果如图 7.17a、b 所示。

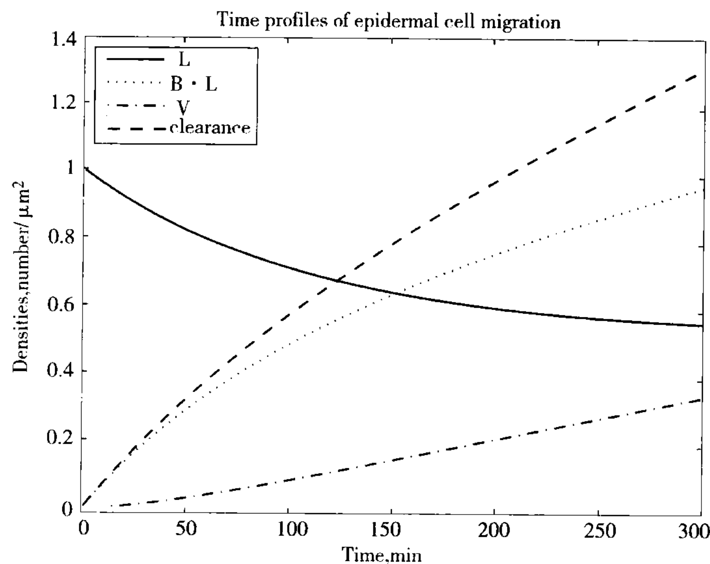


图 7.17a　表皮细胞迁移中各种物质的浓度变化时间曲线

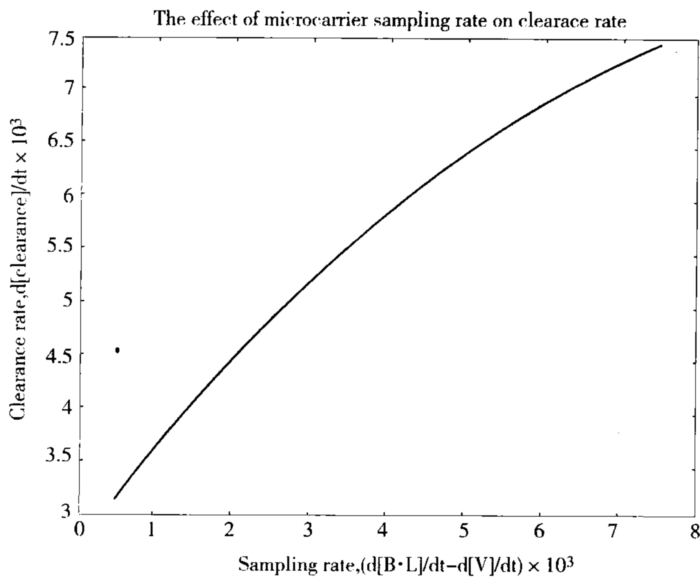


图 7.17b　微载体采样率对于清除率的影响

结果分析：

图 7.17a 显示了配体与皮肤表皮细胞相互作用过程中各种物质的浓度变化曲

线。随着时间的流逝，游离配体的浓度逐渐减小，体现了配体细胞内吞引起的配体浓度损耗。同时，膜受体-配体复合物的浓度 $[B \cdot L]$ 和胞吞配体浓度 $[V]$ 不断增加，配体的清除率也不断增加，表明细胞对于配体微载体还没有饱和。

图 7.17b 显示了细胞清除率随采样率变化的曲线。这里假设胞吞配体对于增强细胞迁移的胞内信号系统没有激活作用。可见，随着配体采样率的增加，细胞清除率单调上升。这表明细胞迁移与配体采样过程密切相关。

7.9 本章学习要点

学习本章之后，读者应该掌握以下内容：

1）用常微分方程可以建立各种生理系统动态过程的仿真模型。

2）常微分方程有如下不同的分类：一阶、二阶、三阶等；线性和非线性；齐次和非齐次；初值问题和边值问题。

3）二阶以及二阶以上的常微分方程可以用本章所述的方法转化为一阶常微分方程组，然后进行数值求解。

4）线性常微分方程组的解由方程组的特征值和特征矢量决定。

5）非线性和线性微分方程组可以用有限差分法求数值积分。

6）求微分方程的数值积分就像爬山，沿着曲线的斜率方向（或者是曲线上不同点斜率的加权平均），仔细选取多个小步长前行，直到到达目的地为止。

7）非线性微分方程的稳定性取决于方程雅可比矩阵的特征值。

8）数值解的稳定性取决于方程的形式、求解算法以及积分步长。

7.10 习题

7.1 假设外加恒定的膜电流，大小为 $10\mu A/cm^2$，请计算 0 ~ 50ms 时间段内 Hodgkin-Huxley 模型的积分（参见例题 7.5），并完整地分析和解释所得到的结果。

7.2 如图 7.18 所示，Enderle 等人（2005）用两房室系统建立了病人体液透析过程的模型。其中，R 为病人的尿素产生速率，V_1 为细胞内液体积，V_2 为细胞外液体积（即血液和细胞间液），C_1 和 C_2 分别为两个房室的尿素浓度，k_{12} 和 k_{21} 分别为两个房室之间的物质传输参数，k_2 为透析装置的尿素清除率常数。

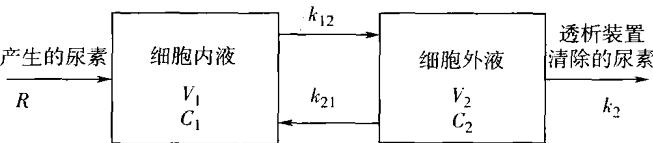


图 7.18 病人体液透析过程的两房室模型

由每个房室尿素的非稳态物质平衡原理可得以下2个微分方程：

$$V_1\frac{dC_1}{dt}=R-k_{12}C_1+k_{21}C_2$$
$$V_2\frac{dC_2}{dt}=k_{12}C_1-k_{21}C_2-k_2C_2 \tag{1}$$

某病人X的参数如下：

$$R=100\text{mg/h}\qquad k_{12}=k_{21}=33\text{L/h}$$
$$V_1=10\text{L}\qquad V_2=25\text{L}$$

透析装置的尿素清除率常数为 $k_2=8\text{L/h}$。

病人X透析之前的血液尿素氮（BUN）浓度为150mg/L。请求解微分方程组（1），并回答以下问题：

（a）要使BUN下降到75mg/L，病人需要进行几小时透析？

（b）透析之后再过多久病人的BUN又会回升到150mg/L？

（c）将 k_{12} 和 k_{21} 设成不同的值，例如，$k_{21}=0.7k_{12}$，也就是降低细胞外液到细胞内液的传输速度。在这种情况下求解方程组，并分析结果。

请说明如何求解，并作图显示各个问题中尿素浓度 C_1 和 C_2 随时间变化的曲线。

7.3　Huang（1994）建立了人体生理性膝跳反射的计算机仿真模型。敲击腿的膝盖骨韧带会产生膝跳反射，随后是钟摆式的振荡。其产生机制是：叩击膝腱时，牵拉肌肉，使其中的感受器发放神经冲动脉冲，传入到脊髓，脊髓的反射信号通过传出神经传回至股四头肌，从而使肌肉突然收缩，使小腿前伸。肌肉松弛之后，小腿就像阻尼钟摆，会来回振荡数次，最后，才回复到正常位置。

Huang在建模时作了如下假设：伸肌和屈肌相同，只是作用方向相反；传入神经和传出神经的末梢数目相等；与膝跳反射系统的响应速度相比，神经信号的传递瞬间完成；小腿偏转角度较小，偏转范围内阻尼系数恒定。根据钟摆方程以及小幅度振荡特性，Huang建立了如下二阶常微分方程：

$$J\frac{d^2\theta}{dt^2}+c\frac{d\theta}{dt}+\left(\frac{mgL\theta}{2}-T\right)\theta=0 \tag{1}$$

式中　θ——膝跳反射过程中小腿的偏转角度；

m——小腿质量；

g——重力加速度常数；

L——小腿长度；

J——小腿惯性力矩；

T——等长肌肉收缩扭矩所产生的增益。则系统的固有频率 ω_n 为

$$\omega_n=\sqrt{\frac{mgL}{2J}-\frac{T}{J}} \tag{2}$$

阻尼因子 α 为

$$\alpha=\frac{c}{2\sqrt{J\left(\frac{mgL}{2}-T\right)}} \tag{3}$$

ω_n 和 α 的值通过实验测得。请用以下给定的 m、g、L、J、ω_n 和 α 的值求解这些方程，并作图显示膝跳反射过程中小腿角度随时间变化的曲线。

$$m = 4\text{kg} \qquad g = 9.81\text{m/s}^2 \qquad L = 0.34\text{m}$$
$$J = 0.154\text{kg}\cdot\text{m}^2 \qquad \omega_n = 6.28\text{rad/s} \qquad \alpha = 0.228$$

用初始条件 $\theta(0) = 0\text{rad}$ 和 $\mathrm{d}\theta(0)/\mathrm{d}t = 2\pi\text{rad/s}$，求解微分方程（1）。提示：用 MATLAB 的 solve 指令求解代数方程，用 dsolve 指令求解微分方程。

7.4　习题 7.2 用两房室系统仿真了病人体液的透析过程，如图 7.19 所示，改用单房室模型，把病人的总体液用一个量 V_T 表示。

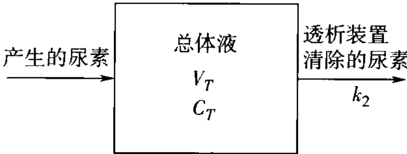


图 7.19　病人体液透析过程的单房室模型

请推导这个单房室模型的尿素的非稳态质量平衡方程。某病人 X 的单房室模型参数如下：

$$R = 100\text{mg/h} \qquad V_T = 35\text{L} \qquad k_2 = 8\text{L/h}$$

病人透析之前的血液尿素氮（BUN）浓度为 150mg/L。请求解该系统的微分方程，并回答以下问题：

（a）要使 BUN 下降到 75mg/L，病人需要进行几小时透析？

（b）透析之后再过多久病人的 BUN 又会回升到 150mg/L？

请将这些结果与习题 7.2 两房室模型得到的结果进行比较。

7.5　图 7.20 所示是一个描述传染病的简单模型。其中，S 为疾病易感人群的人数，I 为感染人群的人数，R 为感染之后已康复（包括死亡）的人数，α 为感染速率常数，β 为康复的速率常数。假设康复病人对传染病产生免疫能力。Edelstein-Keshet（1998）建立了如下描述这 3 个人群之间相互关系的动力学模型：

$$\frac{\mathrm{d}S}{\mathrm{d}t} = -\alpha SI$$
$$\frac{\mathrm{d}I}{\mathrm{d}t} = \alpha SI - \beta I \tag{1}$$
$$\frac{\mathrm{d}R}{\mathrm{d}t} = \beta I$$

假设有一位携带某种传染性极强的新流感病毒的病人，来到一个具有 1000 易感人口的小社区，流感病毒迅速传播，最后，所有易感人口都得了流感。如果这种传染病的速率常数为：

$$\alpha = 0.005\ 人^{-1}\ 周^{-1}$$
$$\beta = 1\ 周^{-1}$$

求解微分方程组（1），并回答以下问题：

（a）传染病达到高峰期需要几周？

（b）在传染病高峰期病人的最大数目是多少？

（c）几周之后传染病衰退（病人人数小于易感人口的 0.5%）？

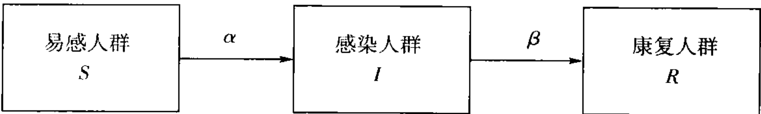


图 7.20　传染病的简单模型

7.6　假设上题传染病模型中的康复病人会失去免疫力，重新成为病毒的易感者（见图 7.21），并且，假设免疫力丧失的速率常数为 $\gamma = 0.1$ 周$^{-1}$，α 和 β 的值与上题相同。求解新模型的微分方程组，并回答以下问题：

（a）传染病达到稳态期需要几周？

（b）传染病稳态期的病人人数有多少？

（c）作出相图，并利用雅可比矩阵的特征值分析解的稳定性。

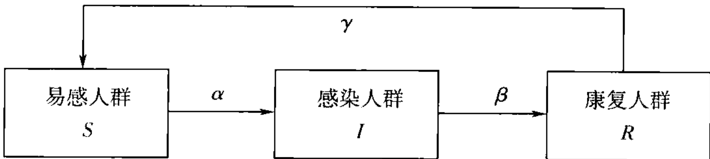


图 7.21　考虑免疫力丧失的传染病改进模型

7.7　著名的范得波（van der Pol）振荡器是如下二阶非线性微分方程：

$$\frac{\mathrm{d}^2 u}{\mathrm{d}t^2} - k(1 - u^2)\frac{\mathrm{d}u}{\mathrm{d}t} + au = 0 \tag{1}$$

方程的解是一个稳定的振荡。范得波用这个方程产生的振荡来模拟心脏搏动等生物节律，并作为心脏起搏器的模型。请用以下 k 值以及初始条件求解范得波方程：

$$k = 1.0\mathrm{s}^{-1} \qquad u(0) = 2 \text{ 无量纲} \qquad \left.\frac{\mathrm{d}u}{\mathrm{d}t}\right|_0 = 0\mathrm{s}^{-1}$$

并确定心律为 1.25 次/s（即 75 次/min，是成人静息状态下的典型心律）时的 a 值。

7.8　众所周知，细菌、干细胞、酵母菌等大多数活细胞可以通过细胞分裂进行自我复制。生物体的生长促使其物质以及所有化学组成有序增长，接着就是细胞分裂为两个相同的子细胞，或者像酵母菌一样，分裂为一个母细胞和一个子细胞。在例题 7.6 中，我们仿真了没有细胞分裂的干细胞分化过程，那个细胞分化模型太简单，因为多数分化阶段都会存在细胞复制活动。细胞复制标志着一个分化阶段的结束和下一个分化阶段的开始。

人体每天要产生和消耗大约 2000 亿个红细胞，骨髓干细胞转化为红细胞的过程称为红细胞生成。从初始阶段的原红细胞开始，分化到完全成熟的无核红细胞大约需要一周时间（Palsson 和 Bhatia，2004）。图 7.22 描述了这个过程。

假设每个细胞离开模型的第（$i-1$）个房室时都先分裂成为两个细胞，再进入第 i 个房室，那么，由质量平衡原理可以导出如下 N 个房室的微分方程：

$$\frac{\mathrm{d}X_1}{\mathrm{d}t} = I - k_1 X_1$$

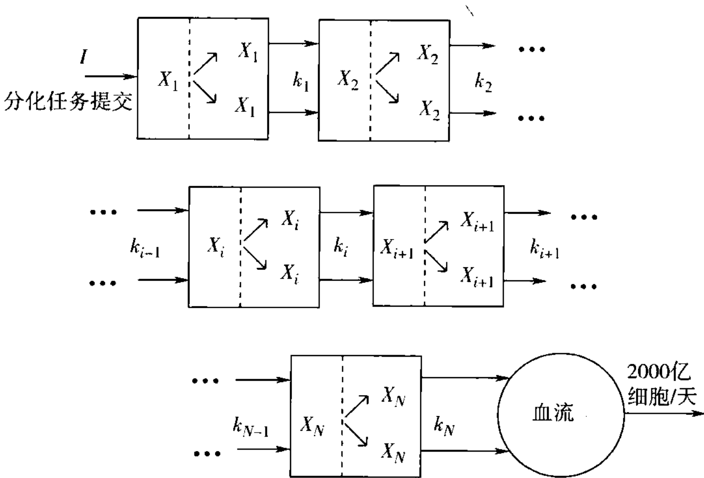


图 7.22　具有复制活动的干细胞分化

$$\frac{dX_2}{dt} = 2k_1X_1 - k_2X_2$$

$$\vdots$$

$$\frac{dX_i}{dt} = 2k_{i-1}X_{i-1} - k_iX_i \tag{1}$$

$$\vdots$$

$$\frac{dX_N}{dt} = 2k_{N-1}X_{N-1} - k_NX_N$$

另外，假设分化过程产生的所有红细胞都进入血流，执行其功能之后死亡，速率为每天 2000 亿个细胞。健康人体内的红细胞数目保持常数不变，也就是处于稳态。于是，血流平衡方程为

$$\frac{dX_{血流}}{dt} = 2k_NX_N - 200 \times 10^9 = 0 \tag{2}$$

用以上微分方程组求红细胞生成的数值仿真结果，并回答以下问题：

（a）分析这些方程的稳态情况，用 I 和 k_i 表示模型每个房室的稳态细胞数目 $X^*(i)$。

（b）假设干细胞分化需要经历 10 个分化阶段（即 $N = 10$），并设初始条件和转化率常数为

$$X_i(0) = 0, 其中\ i = 1, \cdots, N$$
$$k_i = 2.2/天, 其中\ i = 1, \cdots, N$$

为了产生每天所需的 2000 亿个红细胞，每天共需要多少干细胞（I）加入红细胞生成过程？请详细解释如何计算该 I 值。

（c）作图显示这 N 个分化和复制阶段的全部时间曲线，分析结果，并将所得结果与例题 7.6 第 c 种情况的结果进行比较。

7.11 参考文献

Burden, R. L., Faires, J. D., and Reynolds, A. C. 1981. *Numerical Analysis*. Boston, MA: Prindle, Weber & Schmidt.

Constantinides, A., and Mostoufi, N. 1999. *Numerical Methods for Chemical Engineers with MATLAB Applications*. Upper Saddle River, NJ; Prentice Hall PTR.

Edelstein-Keshet, L. 1988. *Mathematical Models in Biology*. New York: McGraw-Hill Book Company.

Enderle, J. D., Blanchard, S. M., and Bronzino, J. D. 2005. *Introduction to Biomedical Engineering*. 2nd ed. San Diego, CA: Academic Press.

Fournier, R. L. 1999. *Basic Transport Phenomena in Biomedical Engineering*. Philadelphia, PA: Taylor & Francis.

Hairer, E., Lubich, C., and Roche, M. 1980. *The Numerical Solution of Differential-Algebraic Systems by Runge-Kutta Methods*. Berlin: Springer-Verlag.

Hairer, E., and Wanner, G. 1991a. *Solving Ordinary Differential Equations I*. Berlin: Springer.

Hairer, E., and Wanner, G. 1991b. *Solving Ordinary Differential Equations II*. Berlin: Springer.

Hodgkin, A. L., and Huxley, A. F. 1952. A Quantitative Description of Membrane Current and its Application to Conduction and Excitation in Nerve. *J. Physiol*,. 117:500-544.

Huang, B. K. 1994. *Computer Simulation Analysis of Biological and Agricultural Systems*. Boca Raton, FL: CRC Press.

Keener, J., and Sneyd, J. (1998). *Mathematical Physiology*. New York: Springer-Verlag.

Kubícek, M. and Hlavàcek, V. 1975. *Numerical Solution of Nonlinear Boundary Value Problems with Applications*. New York: Prentice Hall.

Lauffenburger, D. A., and A.F. Horwitz (1996). Cell Migration: A Physically Integrated Process. *Cell*. 84:359--369.

Palsson, B. Ø., and Bhatia, S. N. 2004. *Tissue Engineering*. Upper Saddle River, NJ: Pearson Prentice Hall.

Tjia, J. S., and Moghe, P. V. 2002a. Regulation of Cell Motility on Polymer Substrates via Dynamic, Cell-Internalizable, Ligand Microinterfaces. *Tissue Eng.*, 8:247-259.

Tjia, J. S. and Moghe, P. V. 2002b. Cell-Internalizable Ligand Microinterfaces on Biomaterials: Design of Regulatory Determinants Of Cell Migration in *Biomimetic Materials and Design: Interactive Biointerfacial Strategies for Drug Delivery and Tissue Engineering*. A. Dillow, T. Lowman (Eds.), Marcel-Dekker, 335-373.

Tjia, J. S., and Moghe, P. V. 2002c. Cell Migration on Cell-Internalizable Ligand Microdepots: A Phenomenological Model. *Annals of Biomedical Engineering*, 30:851-866.

Tortora, G. J., and Grabowski, S. R. 2001. *Introduction to the Human Body*, 5th ed. Hoboken, NJ: John Wiley & Sons, Inc.

第 8 章　动态系统：偏微分方程

8.1　绪论

物质传输是生物系统功能实现的基础，生物体体重的大部分是传输系统的体液，给体内所有组织来来回回运输营养物质和能量。为了分析体内的生理过程和细胞活动过程，生物医学工程人员必须了解物质传输的机制，并且能够求解描述这些机制的数学模型。另外，许多用于诊断和治疗的医学仪器的设计和操作也都需要涉及体液的流动和营养物质的传输。

质量守恒定律、动量守恒定律和能量守恒定律是研究物质传输过程的理论基础，应用这些定律可以建立方程，用于描述传输系统中速度、温度、浓度等变量随时间和地点所发生的变化。这类系统具有多个自变量，其动力学过程需要用偏微分方程（Partial Differential Equation，PDE）来模拟。本章的主要内容就是讲述偏微分方程的数值解法。

生物医学工程中最常见的偏微分方程都是一阶或二阶方程，本章就着重讲述这两类方程。其中，8.2 节列举了生物系统偏微分方程模型的一些例子，8.3 节介绍偏微分方程及其边界条件的分类，以后几节则利用有限差分分析法，建立一阶和二阶偏微分方程数值求解的算法，并用这些方法求解本章提到的几个生理系统模型。

本章主要包括如下学习内容：

1）用偏微分方程建立生理系统的动力学模型；

2）用有限差分近似法表示偏微分方程；

3）求偏微分方程的数值解，作图显示结果，并解释生物系统在不同条件下的动态变化过程；

4）分析模型以及数值解的准确度以及稳定性。

8.2　生物医学系统中的偏微分方程

8.2.1　生物膜的跨膜扩散

细胞的组成大部分为有机化合物和水，水占了人体体重的 60% 以上

(Enderle 等，2005)。化学物质通过细胞膜进出细胞有几种不同的扩散机制（参见7.14节），例如，图8.1的模型说明了葡萄糖和氧气等营养物质的跨膜扩散机制。图中所示是一小块细胞膜，为了简化，假设膜是平的，并且营养物质在细胞外液的浓度较高，因此，图8.1中的扩散自左向右进行。膜的厚度（x 方向）有限，为 L；而膜的长度（y 方向）和宽度（z 方向）相对于厚度而言为无限大。选取膜中的一个微分元（即控制体积），厚为 Δx，其垂直于 x 方向的表面积为 A。下面我们在此控制体积处建立质量平衡方程。非稳态质量平衡方程的一般形式为

$$(\text{物质流入速率}) = (\text{物质流出速率}) + (\text{物质积累速率}) \tag{8.1}$$

进入膜的物质流量由 Fick 第一扩散定律给出

$$J = -D\frac{\mathrm{d}C}{\mathrm{d}x} \tag{8.2}$$

式中　J——流量，单位为 moles/cm^2/s；

C——营养物质的浓度，单位为 moles/cm^3；

D——营养物质在膜中的扩散系数，单位为 cm^2/s。

因此，Fick 定律表明物质的扩散是沿着浓度降低的方向进行的，其扩散速率与浓度梯度成正比，比例系数就是 D。在某时间段 Δt 上将式（8.1）应用于控制体积，可得

$$J_x A = J_{x+\Delta x} A + \frac{C_{x,t+\Delta t} - C_{x,t}}{\Delta t} A\Delta x \tag{8.3}$$

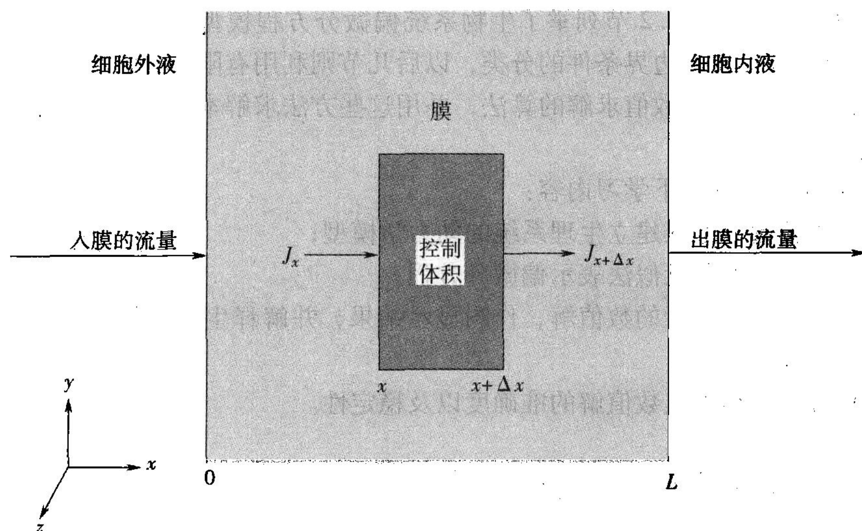


图8.1　营养物质的跨膜扩散

将式（8.2）代入式（8.3），并重排，可得

$$\frac{C_{x,t+\Delta t}-C_{x,t}}{\Delta t}=\frac{\left[D\left.\frac{\mathrm{d}C}{\mathrm{d}x}\right|_{x+\Delta x}-D\left.\frac{\mathrm{d}C}{\mathrm{d}x}\right|_{x}\right]}{\Delta x} \tag{8.4}$$

使 Δx 和 Δt 趋于 0，求式（8.4）的极限，得到如下偏微分方程：

$$\frac{\partial C}{\partial t}=D\frac{\partial^2 C}{\partial x^2} \tag{8.5}$$

该方程的左边是浓度随时间的变化，而右边则是膜内浓度随空间的变化。这就是 Fick 第二扩散定律，是一个一维非稳态偏微分方程，其数值求解方法将在 8.5.2 节讲述，并将用于例题 8.3 的求解。Fick 第二扩散定律的三维形式为

$$\frac{\partial C}{\partial t}=D\left(\frac{\partial^2 C}{\partial x^2}+\frac{\partial^2 C}{\partial y^2}+\frac{\partial^2 C}{\partial z^2}\right) \tag{8.6}$$

用于描述 x、y 和 z 3 个方向都存在扩散的情况。

8.2.2　大分子扩散和药物释放控制

局部用药是医学中的常用方法。有人用 Fick 第二扩散定律等方程模拟了药物从给药基质跑到皮肤表面并进入皮肤的过程（Kubota 等，2002）。假设给药基质的厚度为 L_m，则其中的药物浓度 C_m 可以用如下方程描述：

$$\frac{\partial C_m}{\partial t}=D_m\frac{\partial^2 C_m}{\partial x^2}\qquad -L_m\leqslant x\leqslant 0,\quad t>0 \tag{8.7}$$

设皮肤厚度为 L_s，则其中的药物浓度 C_s 用如下方程描述：

$$\frac{\partial C_s}{\partial t}=D_s\frac{\partial^2 C_s}{\partial x^2}\qquad 0\leqslant x\leqslant L_s,\quad t>0 \tag{8.8}$$

式中　D_m——给药基质的扩散系数；

D_s——皮肤的扩散系数。

这两个方程在给药基质和皮肤的接触边界耦合在一起。利用聚合材料实现经皮给药就是这类问题的一个例子，聚合物可以释放精确控制的大分子药量（Randomsky 等，1990），本章习题 8.5 介绍了经皮给药的模型。

8.2.3　人造血管中的细胞迁移

人造血管材料的设计是组织工程和组织修复领域的重要研究方向。人造血管植入体内用于修复被损伤或闭塞的血管时可能会发生细菌感染，白细胞是抑制急性炎症的关键因素，因此，控制白细胞在人造血管表面的运动对于提高植入假体的抗感染能力非常重要。

Rosenson-Schloss 等人（2002）利用如下扩散对流的方程描述了细胞在人造材料上的运动：

$$\frac{\partial C}{\partial t} = \mu_D \frac{\partial^2 C}{\partial z^2} - v_{eff} \frac{\partial C}{\partial z} \tag{8.9}$$

其初始条件和边界条件为

$$\begin{aligned} &当 t = 0 \quad 且 z > 0 时 \quad C = 0 \\ &当 t > 0 \quad \begin{cases} 若 z = 0 \quad 则 C = C_0 \\ 若 z = z_r \quad 则 C = 0 \end{cases} \end{aligned} \tag{8.10}$$

式中 C——细胞在人造材料表面上的浓度。

式（8.9）右边的第一项与 Fick 第二扩散定律相似，μ_D 是随机迁移系数；但是，方程右边加了第二项，也就是对流传输项，v_{eff} 为细胞的定向迁移速度。这个人造血管问题将在例 8.2 中作进一步讨论。

8.2.4 生理血管和体外管道中的流体流动

生物流体十分复杂，既具有类似固体的性质，又具有类似液体的性质，并且还会随着时间的不同而发生变化。生物流体有血液、润滑关节的滑液、淋巴液、眼球玻璃体中的凝胶液等等（Truskey 等人，2004）。定量分析血流的特性对于认识生理和病理机制、设计和使用血流治疗仪器都非常重要。

最简单的情况下，动脉血流可以看作周期性压力场作用下的刚性圆管中不可压缩牛顿流体的层流，如果血流中只存在速度为 v_z 的轴向流动，那么，可以用如下柱形坐标系统的 Navier-Stokes 方程来描述：

$$\rho \frac{\partial v_z}{\partial t} = -\frac{\partial p}{\partial z} + \frac{\mu}{r} \frac{\partial}{\partial r}\left(r \frac{\partial v_z}{\partial r}\right) \tag{8.11}$$

假设压力梯度周期性变化的频率为 ω，用于模拟心脏的搏动，其方程为

$$-\frac{\partial p}{\partial z} = \frac{\Delta p}{L}\cos(\omega t) \tag{8.12}$$

该模型的求解作为本章的习题 8.6。

8.3 偏微分方程分类

偏微分方程是根据阶数、是否为线性以及边界条件分类的。方程中所出现的最高阶偏导数的阶数就是偏微分方程的阶，下面是一阶、二阶和三阶偏微分方程的例子：

一阶
$$\frac{\partial u}{\partial x} - \alpha \frac{\partial u}{\partial y} = 0 \tag{8.13}$$

二阶
$$\frac{\partial^2 u}{\partial x^2} + u \frac{\partial u}{\partial y} = 0 \tag{8.14}$$

三阶
$$\left(\frac{\partial^3 u}{\partial x^3}\right)^2 + \frac{\partial^2 u}{\partial x \partial y} + \frac{\partial u}{\partial y} = 0 \tag{8.15}$$

偏微分方程可以分为线性、拟线性以及非线性方程。假设有如下二阶方程：

$$a(\cdot)\frac{\partial^2 u}{\partial y^2} + 2b(\cdot)\frac{\partial^2 u}{\partial x \partial y} + c(\cdot)\frac{\partial^2 u}{\partial x^2} + d(\cdot) = 0 \tag{8.16}$$

如果其系数为常数，或者只是自变量的函数，也就是 $(\cdot) \equiv (x,y)$，则式（8.16）是线性方程。如果系数为因变量或者因变量导数的函数，但导数的阶数低于微分方程的阶数，也就是 $(\cdot) \equiv (x,y,u,\partial u/\partial x,\partial u/\partial y)$ 则式（8.16）是拟线性方程。最后，如果系数为因变量导数的函数，且导数的阶数与方程的阶数相同，也就是 $(\cdot) \equiv (x,y,u,\partial^2 u/\partial x^2,\partial^2 u/\partial y^2,\partial^2 u/\partial x \partial y)$，则式（8.16）是非线性方程。根据这些定义，式（8.13）是线性的，式（8.14）是拟线性的，式（8.15）则是非线性的。

两个自变量的线性二阶偏微分方程可以进一步分成 3 种规范类型：椭圆型、抛物型和双曲型。这类方程的一般形式为

$$a\frac{\partial^2 u}{\partial x^2} + 2b\frac{\partial^2 u}{\partial x \partial y} + c\frac{\partial^2 u}{\partial y^2} + d\frac{\partial u}{\partial x} + e\frac{\partial u}{\partial y} + fu + g = 0 \tag{8.17}$$

其中的系数只是常数，或者是自变量的函数。用如下标准区分 3 种不同的规范类型：

$$\text{当 } b^2 - ac < 0 \text{ 时，方程为椭圆型；} \tag{8.18}$$

$$\text{当 } b^2 - ac = 0 \text{ 时，方程为抛物型；} \tag{8.19}$$

$$\text{当 } b^2 - ac > 0 \text{ 时，方程为双曲型。} \tag{8.20}$$

如果 $g=0$，则式（8.17）为齐次微分方程。

分别与这 3 种规范型方程相一致的典型的二阶偏微分方程有：

拉普拉斯方程为椭圆型方程，即

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \tag{8.21}$$

扩散方程，也就是热传导方程，为抛物型方程，即

$$\frac{\partial u}{\partial t} = \alpha\frac{\partial^2 u}{\partial x^2} \tag{8.22}$$

波动方程为双曲型方程，即

$$\frac{\partial^2 u}{\partial t^2} = a^2\frac{\partial^2 u}{\partial x^2} \tag{8.23}$$

下面将证明偏微分方程的求解方法取决于它们的规范类型。由于这些方程的系数可以是自变量的函数，因此，在积分区间 (x,y) 上方程可以从一种规范形式转变成另一种形式。

8.4 初始条件和边界条件

为了求得偏微分方程的惟一数值解，必须给定方程的初始条件和边界条件。偏微分方程的边界条件也可以分成3类，下面利用8.2.1节建立的一维非稳态扩散方程加以说明，该方程为

$$\frac{\partial C}{\partial t}=D\frac{\partial^2 C}{\partial x^2} \tag{8.24}$$

此方程用于描述固体或液体薄片材料（比如细胞膜）中的浓度变化。其中，假设物质传输只发生在 x 方向（见图8.2）。下面是3类边界条件：

Dirichlet 边界条件（即第一类边界条件）：

在固定的自变量值上给定因变量的值。例如，扩散方程的 Dirichlet 类型初始条件可以是

$$C|_{t=0}=f(x) \qquad \text{当 } t=0 \text{ 且 } 0\leqslant x\leqslant 1 \text{ 时}$$

或者

$$C|_{t=0}=C_0 \qquad \text{当 } t=0 \text{ 且 } 0\leqslant x\leqslant 1 \text{ 时}$$

这些初始条件指定了细胞膜内的初始浓度是位移的函数 $f(x)$，或者是常数 C_0（见图8.2a）。Dirichlet 类型边界条件表示为

$$C|_{x=0}=f(t) \qquad \text{当 } x=0 \text{ 且 } t>0 \text{ 时}$$

和

$$C|_{x=1}=C_1 \qquad \text{当 } x=1 \text{ 且 } t>0 \text{ 时}$$

这些边界条件指定了左边界上因变量的值是时间的函数 $f(t)$，比如，这可以指细胞膜左边（胞外）的溶液浓度是根据预先设定的规律变化的；而右边界上因变量的值是常数 C_1，比如，溶液量很大时，可以认为其浓度恒定不变（见图8.2a）。

Neumann 边界条件（即第二类边界条件）：

因变量的导数给定为常数或者自变量的函数。例如：

$$\left.\frac{\partial C}{\partial x}\right|_{x=1}=0 \qquad \text{当 } x=1 \text{ 且 } t>0 \text{ 时}$$

该条件指定了右边界上的浓度梯度为0。在膜扩散问题中，这种情况在理论上等价于细胞膜右边贴上了防渗材料（见图8.2b）。

Cauchy 边界条件：将以上 Dirichlet 边界条件和 Neumann 边界条件结合在一起，就称为 Cauchy 边界条件（见图8.2b）。

Robbins 边界条件（即第三类边界条件）：

因变量的导数给定为因变量自身的函数。对于膜扩散问题，在细胞膜与溶液之间界面上的传输速率可以受到边界层中的对流的控制，于是，传输速率就与界面和溶液之间的浓度差有关，也就是

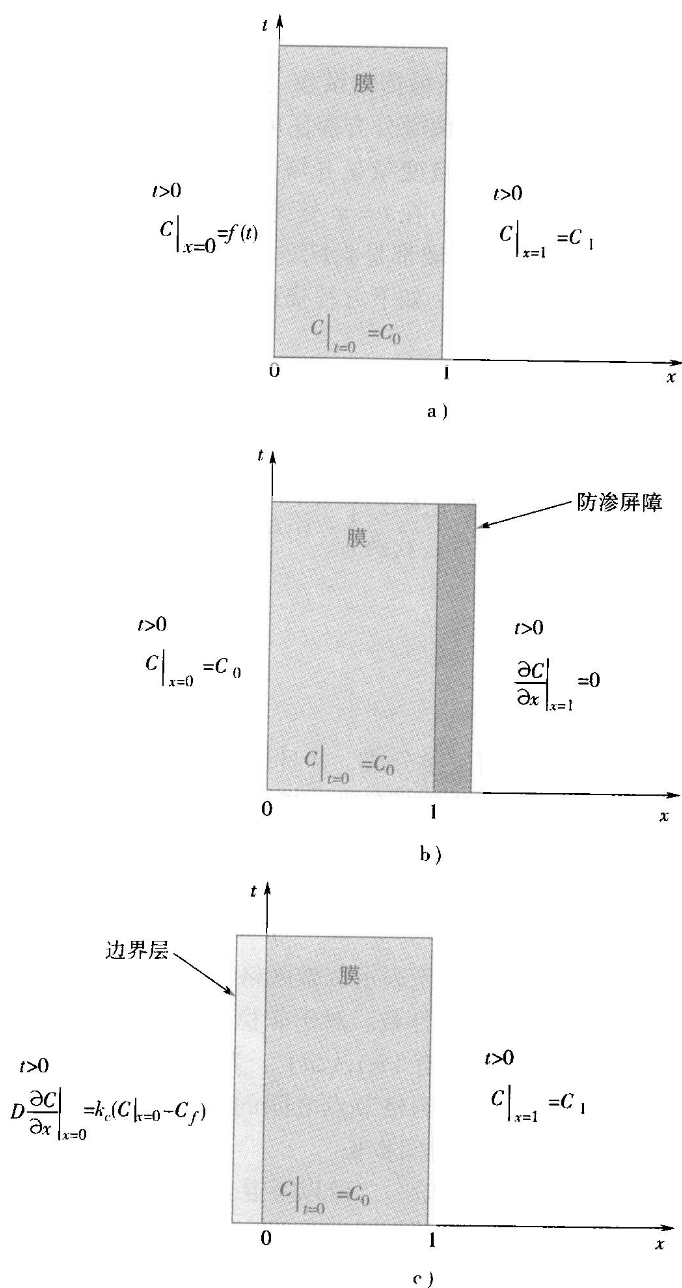


图 8.2　扩散问题的初始条件和边界条件举例

a）Dirichlet 边界条件　b）Cauchy 边界条件（Dirichlet 边界条件和 Neumann 边界条件的结合）

c）左边为 Robbins 边界条件，右边为 Dirichlet 边界条件

$$D\frac{\partial C}{\partial x}\bigg|_{x=0} = k_c(C|_{x=0} - C_f) \quad \text{当}\, x = 0\, \text{且}\, t > 0\, \text{时}$$

式中 k_c——流体界面层的对流质量传输系数（见图 8.2c）。

根据初始条件和边界条件，偏微分方程还可以进一步分为初值问题和边值问题。对于初值问题，至少有一个自变量是开域的。例如，在非稳态扩散问题中，时间变量的取值范围为 $0 \leqslant t \leqslant \infty$，在 $t = \infty$ 处无给定条件，因此，这是一个初值问题。如果所有自变量的取值区域都是封闭的，并且，所有边界上都给定了条件，那么这个问题就是边值问题。如下方程描述的三维稳态热传导问题就是一个边值问题：

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2} = 0 \tag{8.25}$$

所有 6 个界面上的边界条件为

$$\left.\begin{matrix} T(0,y,z) \\ T(1,y,z) \end{matrix}\right\} = \text{给定值} \qquad \left.\begin{matrix} T(x,0,z) \\ T(x,1,z) \end{matrix}\right\} = \text{给定值} \qquad \left.\begin{matrix} T(x,y,0) \\ T(x,y,1) \end{matrix}\right\} = \text{给定值} \tag{8.26}$$

8.5 求解偏微分方程

在第 6 章，我们建立了有限差分法，并且证明了利用有限差分算子代替微分算子可以得到具有任意阶次准确度的导数的近似值。

本节，我们采用同样的处理方法，用有限差分来表示偏导数。偏微分方程包含多个自变量，因此，如图 8.3 所示，我们先分别建立 2 个自变量和 3 个自变量的二维和三维网格。

符号 (i,j) 和 (i,j,k) 分别用于表示二维网格和三维网格的节点，其中，i、j 和 k 分别为 x、y 和 z 方向的节点计数。对于非稳态问题，还有一个时间自变量，用计数器 n 表示时间维，那么就有 (i,j,k,n)。为了尽可能简化表达形式，一般忽略下标，必要时才使用下标。网格节点之间的距离用 Δx、Δy 和 Δz 表示。当时间也是自变量时，用 Δt 表示时间步长。

下面就用有限差分来表示一阶、二阶以及混合偏导数。这里只解释利用中心差分法求近似值的推导过程，有关利用向前差分法和向后差分法得到的结果只在表中列出相应的公式。

u 对于 x 求偏导数，就是意味着 y 和 z 的值保持不变，因此

$$\frac{\partial u}{\partial x}\bigg|_{i,j,k} \equiv \frac{\mathrm{d}u}{\mathrm{d}x}\bigg|_{i,j,k} \tag{8.27}$$

利用中心差分法求一阶导数近似值的公式是式（6.40），将其转化到三维空间，

a）

b）

图 8.3　有限差分网格
a）二维网格　b）三维网格

可得

$$\left.\frac{\partial u}{\partial x}\right|_{i,j,k} = \frac{1}{2\Delta x}(u_{i+1,j,k} - u_{i-1,j,k}) + O(\Delta x^2) \tag{8.28}$$

同理，y 和 z 方向的一阶偏导数为

$$\left.\frac{\partial u}{\partial y}\right|_{i,j,k} = \frac{1}{2\Delta y}(u_{i,j+1,k} - u_{i,j-1,k}) + O(\Delta y^2) \tag{8.29}$$

$$\left.\frac{\partial u}{\partial z}\right|_{i,j,k} = \frac{1}{2\Delta z}(u_{i,j,k+1} - u_{i,j,k-1}) + O(\Delta z^2) \tag{8.30}$$

同样，用式（6.41）表示的二阶中心差分偏导数为

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{i,j,k} = \frac{1}{\Delta x^2}(u_{i+1,j,k} - 2u_{i,j,k} + u_{i-1,j,k}) + O(\Delta x^2) \tag{8.31}$$

$$\left.\frac{\partial^2 u}{\partial y^2}\right|_{i,j,k} = \frac{1}{\Delta y^2}(u_{i,j+1,k} - 2u_{i,j,k} + u_{i,j-1,k}) + O(\Delta y^2) \tag{8.32}$$

$$\left.\frac{\partial^2 u}{\partial z^2}\right|_{i,j,k} = \frac{1}{\Delta z^2}(u_{i,j,k+1} - 2u_{i,j,k} + u_{i,j,k-1}) + O(\Delta z^2) \tag{8.33}$$

最后，混合偏导数为

$$\left.\frac{\partial^2 u}{\partial y \partial x}\right|_{i,j,k} = \frac{\partial}{\partial y}\left[\left.\frac{\partial u}{\partial x}\right|_{i,j,k}\right] \tag{8.34}$$

在 (i,j,k) 处对 $\partial u/\partial x$ 实行 $\partial/\partial y$ 运算等价于在 $(i,j+1,k)$ 和 $(i,j-1,k)$ 两点计算 $\partial u/\partial x$ 值，因此

$$\left.\frac{\partial^2 u}{\partial y \partial x}\right|_{i,j,k}$$

$$= \frac{1}{2\Delta y}\left[\frac{1}{2\Delta x}(u_{i+1,j+1,k} - u_{i-1,j+1,k}) - \frac{1}{2\Delta x}(u_{i+1,j-1,k} - u_{i-1,j-1,k})\right] + O(\Delta x^2 + \Delta y^2)$$

$$= \frac{1}{4\Delta x \Delta y}(u_{i+1,j+1,k} - u_{i-1,j+1,k} - u_{i+1,j-1,k} + u_{i-1,j-1,k}) + O(\Delta x^2 + \Delta y^2) \tag{8.35}$$

表 8.1 归纳了以上偏导数的中心差分近似公式。表 8.2 和表 8.3 则给出了相应的向前差分和向后差分的近似公式。至于选择哪一种差分来表示偏导数，取决于被求解问题的特性、模拟对象的几何形状、以及边界条件的类型。由第 6 章的讨论可知，中心差分法的准确度比向前差分法和向后差分法都要高，因此，中心差分法将是我们的首选。但是，有时使用中心差分法得到的数值解会不稳定（参见 8.5.2 节），这时就需要用向前差分法或向后差分法。另外，如果边界条件是 Neumann 类型或者 Robbins 类型的，根据系统的几何特性，可能用向前差分法或向后差分法表示偏导数更合适（参见 8.5.1 节）。

表 8.1　偏导数的中心差分近似公式

偏　导　数	中 心 差 分	误　　差
$\left.\frac{\partial u}{\partial x}\right\|_{i,j,k}$	$\frac{1}{2\Delta x}(u_{i+1,j,k} - u_{i-1,j,k})$	$O(\Delta x^2)$
$\left.\frac{\partial u}{\partial y}\right\|_{i,j,k}$	$\frac{1}{2\Delta y}(u_{i,j+1,k} - u_{i,j-1,k})$	$O(\Delta y^2)$
$\left.\frac{\partial u}{\partial z}\right\|_{i,j,k}$	$\frac{1}{2\Delta z}(u_{i,j,k+1} - u_{i,j,k-1})$	$O(\Delta z^2)$
$\left.\frac{\partial^2 u}{\partial x^2}\right\|_{i,j,k}$	$\frac{1}{\Delta x^2}(u_{i+1,j,k} - 2u_{i,j,k} + u_{i-1,j,k})$	$O(\Delta x^2)$

（续）

偏导数	中心差分	误差
$\left.\frac{\partial^2 u}{\partial y^2}\right\|_{i,j,k}$	$\frac{1}{\Delta y^2}(u_{i,j+1,k} - 2u_{i,j,k} + u_{i,j-1,k})$	$O(\Delta y^2)$
$\left.\frac{\partial^2 u}{\partial z^2}\right\|_{i,j,k}$	$\frac{1}{\Delta z^2}(u_{i,j,k+1} - 2u_{i,j,k} + u_{i,j,k-1})$	$O(\Delta z^2)$
$\left.\frac{\partial^2 u}{\partial y \partial x}\right\|_{i,j,k}$	$\frac{1}{4\Delta x \Delta y}(u_{i+1,j+1,k} - u_{i-1,j+1,k} - u_{i+1,j-1,k} + u_{i-1,j-1,k})$	$O(\Delta x^2 + \Delta y^2)$

表 8.2　偏导数的向前差分近似公式

偏导数	向前差分	误差
$\left.\frac{\partial u}{\partial x}\right\|_{i,j,k}$	$\frac{1}{\Delta x}(u_{i+1,j,k} - u_{i,j,k})$	$O(\Delta x)$
$\left.\frac{\partial u}{\partial y}\right\|_{i,j,k}$	$\frac{1}{\Delta y}(u_{i,j+1,k} - u_{i,j,k})$	$O(\Delta y)$
$\left.\frac{\partial u}{\partial z}\right\|_{i,j,k}$	$\frac{1}{\Delta z}(u_{i,j,k+1} - u_{i,j,k})$	$O(\Delta z)$
$\left.\frac{\partial^2 u}{\partial x^2}\right\|_{i,j,k}$	$\frac{1}{\Delta x^2}(u_{i+2,j,k} - 2u_{i+1,j,k} + u_{i,j,k})$	$O(\Delta x)$
$\left.\frac{\partial^2 u}{\partial y^2}\right\|_{i,j,k}$	$\frac{1}{\Delta y^2}(u_{i,j+2,k} - 2u_{i,j+1,k} + u_{i,j,k})$	$O(\Delta y)$
$\left.\frac{\partial^2 u}{\partial z^2}\right\|_{i,j,k}$	$\frac{1}{\Delta z^2}(u_{i,j,k+2} - 2u_{i,j,k+1} + u_{i,j,k})$	$O(\Delta z)$
$\left.\frac{\partial^2 u}{\partial y \partial x}\right\|_{i,j,k}$	$\frac{1}{\Delta x \Delta y}(u_{i+1,j+1,k} - u_{i,j+1,k} - u_{i+1,j,k} + u_{i,j,k})$	$O(\Delta x + \Delta y)$

表 8.3　偏导数的向后差分近似公式

偏导数	向后差分	误差
$\left.\frac{\partial u}{\partial x}\right\|_{i,j,k}$	$\frac{1}{\Delta x}(u_{i,j,k} - u_{i-1,j,k})$	$O(\Delta x)$
$\left.\frac{\partial u}{\partial y}\right\|_{i,j,k}$	$\frac{1}{\Delta y}(u_{i,j,k} - u_{i,j-1,k})$	$O(\Delta y)$
$\left.\frac{\partial u}{\partial z}\right\|_{i,j,k}$	$\frac{1}{\Delta z}(u_{i,j,k} - u_{i,j,k-1})$	$O(\Delta z)$
$\left.\frac{\partial^2 u}{\partial x^2}\right\|_{i,j,k}$	$\frac{1}{\Delta x^2}(u_{i,j,k} - 2u_{i-1,j,k} + u_{i-2,j,k})$	$O(\Delta x)$

（续）

偏导数	向后差分	误差
$\left.\frac{\partial^2 u}{\partial y^2}\right\|_{i,j,k}$	$\frac{1}{\Delta y^2}(u_{i,j,k} - 2u_{i,j-1,k} + u_{i,j-2,k})$	$O(\Delta y)$
$\left.\frac{\partial^2 u}{\partial z^2}\right\|_{i,j,k}$	$\frac{1}{\Delta z^2}(u_{i,j,k} - 2u_{i,j,k-1} + u_{i,j,k-2})$	$O(\Delta z)$
$\left.\frac{\partial^2 u}{\partial y \partial x}\right\|_{i,j,k}$	$\frac{1}{\Delta x \Delta y}(u_{i,j,k} - u_{i,j-1,k} - u_{i-1,j,k} + u_{i-1,j-1,k})$	$O(\Delta x + \Delta y)$

如果使用更高准确度的有限差分近似公式，例如，中心差分法的式（6.44）和式（6.45），向前差分法的式（6.31）和式（6.32），以及向后差分法的式（6.18）和式（6.19），相应地就可以得到具有较高准确度的偏导数公式。但是，由于包含的计算项很多，需要的计算时间会很长，因此，高准确度的公式并不常用。

本章后面几节讲述应用有限差分近似法求解椭圆型、抛物型和双曲型偏微分方程的方法。

8.5.1 椭圆型偏微分方程

在稳态扩散和热传导问题中经常遇到椭圆型偏微分方程。例如，描述二维稳态扩散的 Fick 第二定律［即式（8.6）］可以简化为

$$\frac{\partial^2 C}{\partial x^2} + \frac{\partial^2 C}{\partial y^2} = 0 \tag{8.36}$$

这种形式的偏微分方程称为拉普拉斯方程。下面讨论椭圆型偏微分方程的数值解法，先看这个二维稳态扩散问题的一般形式

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \tag{8.37}$$

用中心差分近似公式（8.31）和式（8.32）替换两个二阶偏导数，可得

$$\frac{1}{\Delta x^2}(u_{i+1,j,k} - 2u_{i,j,k} + u_{i-1,j,k}) + \frac{1}{\Delta y^2}(u_{i,j+1,k} - 2u_{i,j,k} + u_{i,j-1,k}) = 0 \tag{8.38}$$

重排成为

$$-2\left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}\right)u_{i,j} + \left(\frac{1}{\Delta x^2}\right)u_{i+1,j} + \left(\frac{1}{\Delta x^2}\right)u_{i-1,j} + \left(\frac{1}{\Delta y^2}\right)u_{i,j+1} + \left(\frac{1}{\Delta y^2}\right)u_{i,j-1} = 0 \tag{8.39}$$

这是包含了 5 个邻近网格节点上因变量的值的线性代数方程。

如图 8.4 所示，一个长为 L 宽为 W 的矩形对象，如果在 x 方向分成 n_x 段，在 y 方向分成 n_y 段，整个矩形共有 $(n_x + 1) \times (n_y + 1)$ 个节点，其中的

$(n_x-1)\times(n_y-1)$ 个是内部节点。对每个内部节点写出方程（8.39），则得到一个由 $(n_x-1)\times(n_y-1)$ 个线性代数方程构成的方程组，其中包含有 $(n_x+1)\times(n_y+1)-4$ 个未知数（4 个角上的节点不出现在方程组中）。该方程组求解所需的其他信息由边界条件提供。如果边界条件是 Dirichlet 类型的，那么因变量在所有边界节点上的值已知；如果边界节点上的边界条件有 Neumann 类型或 Robbins 类型的，只是给定了边界上的偏导数值，那么这些边界条件本身也必须用有限差分的近似值来计算，下面说明这一点。

假设 Neumann 类型和 Robbins 类型的边界条件具有如下统一的形式：

$$\frac{\partial u}{\partial x}=\beta+\gamma u \tag{8.40}$$

当 $\gamma=0$ 时，该式为 Neumann 边界条件；当 $\gamma\neq 0$ 时，该式为 Robbins 边界条件。β 可以是任意数值，包括 0。用有限差分近似值替代式中的偏导数，就可以求得边界条件。如果用中心差分法，则式（8.40）成为

$$\frac{1}{2\Delta x}(u_{i+1,j}-u_{i-1,j})=\beta+\gamma u_{i,j} \tag{8.41}$$

这个近似公式只是在具有 Neumann 条件或 Robbins 条件的边界上才成立。但是，如果在边界上采用中心差分法，就要引入处于对象之外的虚拟节点，因而使问题的求解复杂化。而在边界上采用向前差分法或向后差分法就不会引入虚拟节点。在边界上使用的向前或向后差分法应该与替代微分方程导数的有限差分近似法具有相同的准确度阶数。因此，在这里应该用一阶导数的 $O(h^2)$ 阶向前或向后差分公式［分别为式（6.31）和式（6.18）］。下面说明这些公式在矩形对象的 4 条边界上的用法。

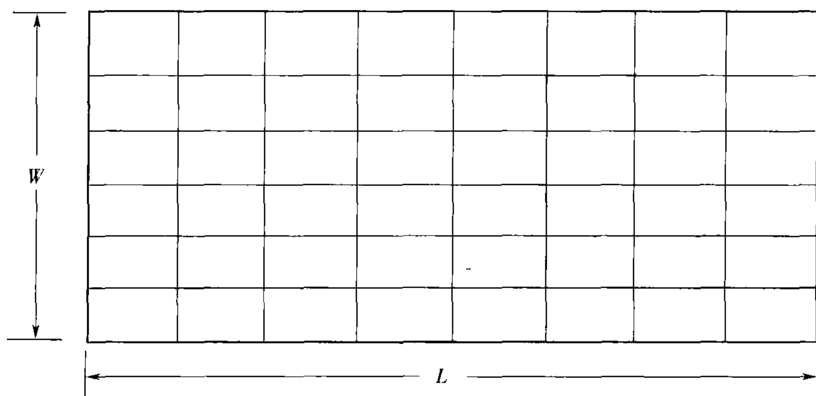


图 8.4　一个矩形对象网格，$n_x=8$，$n_y=6$（共有 63 个节点，其中 35 个是内部节点）

在 x 的下边界，$x=0$ 且 $i=1$。用向前有限差分［即式（6.31）］替换

Neumann或 Robbins 条件即［式（8.40）］中的导数，并且，解得如下 $u_{i,j}$的值：

$$u_{i,j} = \frac{4u_{i+1,j} - u_{i+2,j} - 2\Delta x\beta}{(3 + 2\Delta x\gamma)},\text{其中} \quad i = 1 \tag{8.42}$$

该式只在 x 的下边界存在 Neumann 或 Robbins 条件时才成立。

在 x 的上边界，$x = L$ 且 $i = n_x + 1$。用向后有限差分［即式（6.18）］替换 Neumann 或 Robbins 条件中的导数，并且，解得如下 $u_{i,j}$的值：

$$u_{i,j} = \frac{4u_{i-1,j} - u_{i-2,j} + 2\Delta x\beta}{(3 - 2\Delta x\gamma)},\text{其中} \quad i = n_x + 1 \tag{8.43}$$

式（8.43）只在 x 的上边界存在 Neumann 或 Robbins 条件时才成立。

在 y 的下边界，$y = 0$ 且 $j = 1$。用向前有限差分［即式（6.31）］替换 Neumann 或 Robbins 条件中的导数，并且，解得如下 $u_{i,j}$的值：

$$u_{i,j} = \frac{4u_{i,j+1} - u_{i,j+2} - 2\Delta y\beta}{(3 + 2\Delta y\gamma)},\text{其中} \quad j = 1 \tag{8.44}$$

式（8.44）只在 y 的下边界存在 Neumann 或 Robbins 条件时才成立。

在 y 的上边界，$y = W$ 且 $j = n_y + 1$。用向后有限差分［即式（6.18）］替换 Neumann 或 Robbins 条件中的导数，并且，解得如下 $u_{i,j}$的值：

$$u_{i,j} = \frac{4u_{i,j-1} - u_{i,j-2} + 2\Delta y\beta}{(3 - 2\Delta y\gamma)},\text{其中} \quad j = n_y + 1 \tag{8.45}$$

该式只在 y 的上边界存在 Neumann 或 Robbins 条件时才成立。

式（8.39）及其相应的边界条件组成了一个线性代数方程组，用高斯法可以求解这个方程组。而且，式（8.39）是一个“边界线”对角占优系统，因此，可以用高斯-赛德尔（Gauss-Seidel）法（参见4.5.2节）求解这个问题。重排式（8.39）求得如下 $u_{i,j}$的值：

$$u_{i,j} = \frac{\frac{1}{\Delta x^2}(u_{i+1,j} + u_{i-1,j}) + \frac{1}{\Delta y^2}(u_{i,j+1} + u_{i,j-1})}{2\left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}\right)} \tag{8.46}$$

式（8.46）可以用于高斯-赛德尔迭代法的迭代计算。所有 $u_{i,j}$都要有初始估计值，通过 Dirichlet 边界条件的平均计算可以方便地求得这些值。

如果使用等距网格，即 $\Delta x = \Delta y$，则式（8.46）可以简化为

$$u_{i,j} = \frac{u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}}{4} \tag{8.47}$$

也就是拉普拉斯方程中节点 (i,j) 处因变量的值是该节点上下左右4个节点值的算术平均。图8.5显示了这种计算模式，有时被称为“5点格式”。

同理，利用三维空间的有限差分近似法可以将如下三维椭圆型偏微分方程转化为线性代数方程。

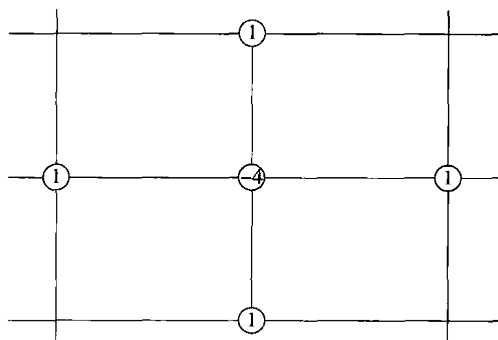


图 8.5　等距网格拉普拉斯的计算模式
[每个圆圈中的数值是微分方程（8.47）中该节点的系数]

$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2}=0 \tag{8.48}$$

将式（8.31）~式（8.33）代入式（8.48），可得

$$\begin{aligned}&\frac{1}{\Delta x^2}(u_{i+1,j,k}-2u_{i,j,k}+u_{i-1,j,k})+\frac{1}{\Delta y^2}(u_{i,j+1,k}-2u_{i,j,k}+u_{i,j-1,k})\\&+\frac{1}{\Delta z^2}(u_{i,j,k+1}-2u_{i,j,k}+u_{i,j,k-1})=0\end{aligned} \tag{8.49}$$

对于等距网格，即 $\Delta x=\Delta y=\Delta z$，该式可以简化为

$$u_{i,j,k}=\frac{u_{i+1,j,k}+u_{i-1,j,k}+u_{i,j+1,k}+u_{i,j-1,k}+u_{i,j,k+1}+u_{i,j,k-1}}{6} \tag{8.50}$$

与二维的情况相同，节点 (i,j,k) 处因变量的值是该节点的 6 个邻近节点值的算术平均。图 8.6 显示了三维椭圆型偏微分方程的计算模式。

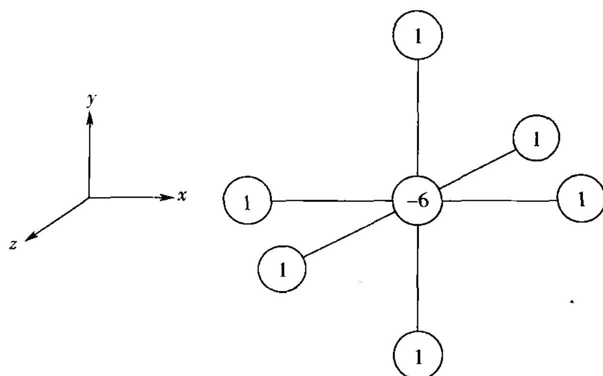


图 8.6　等距网格三维椭圆型偏微分方程的计算模式
[每个圆圈中的数值是微分方程（8.50）中该节点的系数]

非齐次拉普拉斯方程就是泊松方程，即

$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}=f(x,y) \tag{8.51}$$

该方程也属于椭圆型偏微分方程。以下这个泊松方程

$$\frac{\partial^2 \phi}{\partial x^2}+\frac{\partial^2 \phi}{\partial y^2}=-\frac{p}{T} \tag{8.52}$$

可用于描述在均匀压力 p 的作用下，张力为 T 的拉伸膜的位移 ϕ。泊松方程的有限差分公式为

$$-2\left(\frac{1}{\Delta x^2}+\frac{1}{\Delta y^2}\right)u_{i,j}+\left(\frac{1}{\Delta x^2}\right)u_{i+1,j}+\left(\frac{1}{\Delta x^2}\right)u_{i-1,j}+\left(\frac{1}{\Delta y^2}\right)u_{i,j+1}+\left(\frac{1}{\Delta y^2}\right)u_{i,j-1}=f_{i,j} \tag{8.53}$$

由式（8.53）可求得 $u_{i,j}$为

$$u_{i,j}=\frac{\frac{1}{\Delta x^2}(u_{i+1,j}+u_{i-1,j})+\frac{1}{\Delta y^2}(u_{i,j+1}+u_{i,j-1})}{2\left(\frac{1}{\Delta x^2}+\frac{1}{\Delta y^2}\right)}-\frac{f_{i,j}}{2\left(\frac{1}{\Delta x^2}+\frac{1}{\Delta y^2}\right)} \tag{8.54}$$

下面的例题8.1说明了拉普拉斯（泊松）椭圆型偏微分方程的数值求解方法。

例8.1 拉普拉斯（泊松）方程的求解——张力和压力作用下的膜动力学。

问题陈述：

耳蜗是内耳的一部分，是一个充满液体的小室，其中包含了将声音信号转化为神经信号的生理结构。耳蜗由基底膜分成两个腔，上腔为前庭，下腔为鼓阶。耳道中的外部声音压力可以使基底膜产生位移。如下泊松方程描述了在张力 T 和均匀压力 p 作用下的膜的位移 ϕ：

$$\frac{\partial^2 \phi}{\partial x^2}+\frac{\partial^2 \phi}{\partial y^2}=-\frac{p}{T} \tag{8.52}$$

假设膜的面积为 1cm^2，膜的四边都牢固地固定在支架上，因此，周边上没有位移，即

$$\phi(0,y)=0,\quad \phi(1,y)=0,\quad \phi(x,0)=0,\quad \phi(x,1)=0$$

请用以下参数值求膜的位移 ϕ：

$$\frac{p}{T}=0.5\text{cm}^{-1}$$

并用三维图形作图显示计算结果。

解：

下列程序利用高斯-赛德尔迭代法以及 Dirichlet、Neumann 或 Robbins 类型边界条件，计算二维拉普拉斯（泊松）方程［即式（8.46）或式（8.54）］的有

限差分数值解。Neumann 和 Robbins 类型的边界条件用式（8.42）～式（8.45）求值。

```
% example8_1.m-This program solves the two-dimensional
% Laplace or Poisson equation with Dirichlet, Neumann,
% or Robbins boundary conditions. The program uses finite
% differences and the Gauss-Seidel method. It is applied to
% the calculation of the displacement of a stretched membrane.

clc; clear all;
bcdialog=['Lower x boundary condition:'
          'Upper x boundary condition:'
          'Lower y boundary condition:'
          'Upper y boundary condition:'];
bc=zeros(4,3);  % zero the boundary conditions
disp('        Solution of the Laplace or Poisson equation')
disp('(Two-dimensional elliptic partial differential equation)')
disp(' '); disp(' ')
disp('                    Upper y boundary          '   )
disp('          ______________________________      '   )
disp('          |                             |'        )
disp('          |                             |'        )
disp('          |                             |'        )
disp('          |                             |'        )
disp('Lower x   |                             |Upper  x' )
disp('boundary  |                             | boundary')
disp('          |                             |'        )
disp('          |                             |'        )
disp('          y                             |'        )
disp('          |__x______ length (L) ________|'        )
disp('          0                                '      )
disp('                    Lower y boundary          '   )
disp('');
L=input('Length of the object (x-direction) (cm) =');
W=input('Width of the object  (y-direction) (cm) =');
```

```
nx = input(' Number of divisions in x-direction = ');
ny = input(' Number of divisions in y-direction = ');
disp('');
f = input('Right-hand side of the Poisson equation (f) = ');
disp(''); guess = input('Initial starting guess = ');
disp(''); disp(' Boundary conditions:')
for k = 1:4
  disp('')
  disp(bcdialog(k,:))
  disp(' 1 - Dirichlet')
  disp(' 2 - Neumann')
  disp(' 3 - Robbins')
  bc(k,1) = input(' Enter your choice:');
  if bc(k,1) ==1
        bc(k,2) = input(' Value of Dirichlet boundary = ');
  end
  if bc(k,1) ==2
        bc(k,2) = input(' Value of Neummann boundary = ');
  end
  if bc(k,1) ==3
        disp(' Value of Robbins boundary:');
        disp(' u'' = (beta) + (gamma) * u')
        bc(k,2) = input(' Constant  (beta) = ');
        bc(k,3) = input(' Coefficient (gamma) = ');
    end
end
dx = L/nx;  dy = W/ny;     % Calculate the increments
x = [0:nx] * dx; y = [0:ny] * dy;
u = guess * ones(nx + 1,ny + 1); % Set initial guess for all values of u
U = u;
count = 0; iter = 0;          % Zero the counters
total_points = (nx + 1) * (ny + 1);
while count < (total_points)
count = 0; iter = iter + 1;
% Set the boundary conditions
```

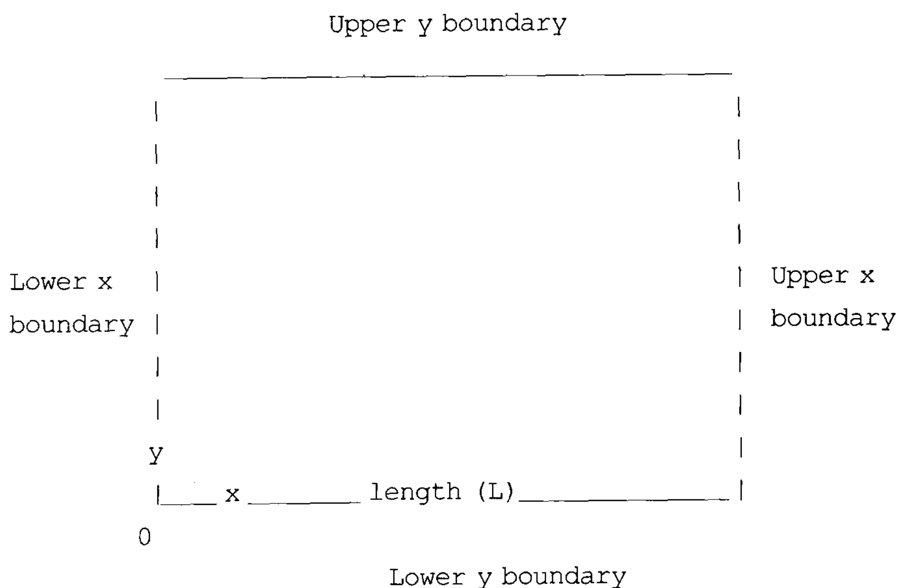
```
% Lower x boundary condition
i =1;
switch bc(1,1)
case 1
    u(i,:) =bc(1,2);
case {2, 3}
    u(i,:) = (4 * u(i +1,:) - u(i +2,:)-2 * dx * bc(1,2))/(3 +2 * dx *
bc(1,3));
end
% Upper x boundary condition
i =nx +1;
switch bc(2,1)
case 1
    u(i,:) =bc(2,2);
case{2, 3}
    u(i,:) = (4 * u(i -1,:) -u(i -2,:) +2 * dx * bc(2,2))/(3 -2 * dx *
bc(2,3));
end
% Lower y boundary condition
j =1;
switch bc(3,1)
case 1
    u(:,j) =bc(3,2);
case{2, 3}
    u(:,j) = (4 * u(:,j +1) -u(:,j +2) -2 * dy * bc(3,2))/(3 +2 * dy *
bc(3,3));
end
% Upper y boundary condition
j =ny +1;
switch bc(4,1)
case 1
    u(:,j) =bc(4,2);
case{2, 3}
    u(:,j) = (4 * u(:,j -1) -u(:,j -2) +2 * dy * bc(4,2))/(3 -2 * dy *
bc(4,3));
```

```
end
% Evaluate all internal points using Gauss-Seidel method
for i =2:nx
    for j =2:ny
        u(i,j) = ((u(i +1,j) +u(i -1,j))/dx^2 + (u(i,j +1)...
            +u(i,j -1))/dy^2 - f)/(2 * ((1/dx^2) + (1/dy^2)));
    end
end
% Examine convergence
for i =1:nx +1
    for j =1:ny +1
        if u(i,j) ~=0
            if abs((U(i,j) - u(i,j))/u(i,j)) <=1e - 12
                count =count +1;
            end
        else
            if abs((U(i,j) - u(i,j))) <=1e-12
                count =count +1;
            end
        end
    end
end
U =u;
end

fprintf('\n Iterations =% g  ',iter)
fprintf('\n Points converged =% g/% g \n\n', count, total _ points)
disp(' The matrix of results is:')
u'
% Plot the final results
clf; figure(1); surf(x,y,u')
xlabel('Length (cm)'); ylabel('Width (cm)'); zlabel('Displacement
(cm)')
shading interp
title('Displacement of the Stretched Membrane')
```

输入：

```
>> example8_1
              Solution of the Laplace or Poisson equation
(Two-dimensional elliptic partial differential equation)
```




```
Length of the object (x-direction) (cm) =1
Width of the object  (y-direction) (cm) =1
Number of divisions in x-direction =10
Number of divisions in y-direction =10

Right-hand side of the Poisson equation (f) = -0.5

Initial starting guess =0.02

Boundary conditions:

Lower x boundary condition:
1 - Dirichlet
2 - Neumann
```

```
3 - Robbins
Enter your choice:1
Value of Dirichlet boundary = 0

Upper x boundary condition:
1 - Dirichlet
2 - Neumann
3 - Robbins
Enter your choice:1
Value of Dirichlet boundary = 0

Lower y boundary condition:
1 - Dirichlet
2 - Neumann
3 - Robbins
Enter your choice:1
Value of Dirichlet boundary = 0

Upper y boundary condition:
1 - Dirichlet
2 - Neumann
3 - Robbins
Enter your choice:1
Value of Dirichlet boundary = 0
```

输出结果:

```
Iterations = 240
 Points converged = 121/121
 The matrix of results is:
ans =
0      0      0      0      0      0      0      0      0      0 0
0 0.0064 0.0103 0.0127 0.0140 0.0144 0.0140 0.0127 0.0103 0.0064 0
0 0.0103 0.0171 0.0215 0.0239 0.0247 0.0239 0.0215 0.0171 0.0103 0
0 0.0127 0.0215 0.0272 0.0304 0.0314 0.0304 0.0272 0.0215 0.0127 0
0 0.0140 0.0239 0.0304 0.0341 0.0353 0.0341 0.0304 0.0239 0.0140 0
```

```
0 0.0144 0.0247 0.0314 0.0353 0.0365 0.0353 0.0314 0.0247 0.0144 0
0 0.0140 0.0239 0.0304 0.0341 0.0353 0.0341 0.0304 0.0239 0.0140 0
0 0.0127 0.0215 0.0272 0.0304 0.0314 0.0304 0.0272 0.0215 0.0127 0
0 0.0103 0.0171 0.0215 0.0239 0.0247 0.0239 0.0215 0.0171 0.0103 0
0 0.0064 0.0103 0.0127 0.0140 0.0144 0.0140 0.0127 0.0103 0.0064 0
0      0      0      0      0      0      0      0      0      0 0
>>
```

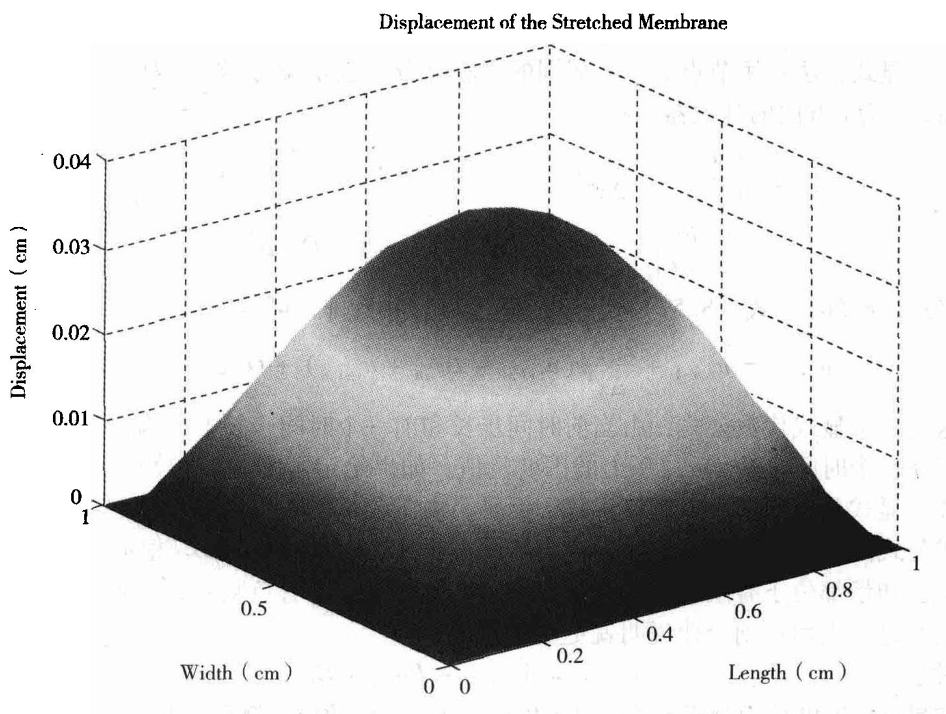


图 8.7　拉伸膜的位移

结果讨论：

以上结果表明，膜的中心部分位移最大。注意：以上程序求解的是函数 f 为 0 时的拉普拉斯方程。

8.5.2 抛物型偏微分方程

抛物型偏微分方程的典型例子是一维非稳态扩散方程（即 Fick 第二扩散定律）

$$\frac{\partial C}{\partial t} = D\frac{\partial^2 C}{\partial x^2} \tag{8.24}$$

还有一维非稳态热传导方程

$$\frac{\partial T}{\partial t} = \alpha\frac{\partial^2 T}{\partial x^2} \tag{8.55}$$

这些方程可以有 Dirichlet、Neumann 和 Cauchy 类型的边界条件。

假设这种一维方程的一般形式为

$$\frac{\partial u}{\partial t} = \alpha\frac{\partial^2 u}{\partial x^2} \tag{8.22}$$

下面介绍用有限差分求解该方程的几种方法。

显式方法：用节点 (i,n) 周围的中心差分来表示偏导数，i 为 x 方向的计数器，n 为 t 方向的计数器，则

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{i,n} = \frac{1}{\Delta x^2}(u_{i+1,n} - 2u_{i,n} + u_{i-1,n}) + O(\Delta x^2) \tag{8.56}$$

$$\left.\frac{\partial u}{\partial t}\right|_{i,n} = \frac{1}{2\Delta t}(u_{i,n+1} - u_{i,n-1}) + O(\Delta t^2) \tag{8.57}$$

将式（8.56）、式（8.57）代入式（8.22），并重排，可得

$$u_{i,n+1} = u_{i,n-1} + \frac{2\alpha\Delta t}{\Delta x^2}(u_{i+1,n} - 2u_{i,n} + u_{i-1,n}) + O(\Delta x^2 + \Delta t^2) \tag{8.58}$$

这是一个显式代数公式，由当前时间步长和前一个时间步长上的因变量的值，计算下一个时间步长（$u_{j,n+1}$）上的因变量值。如果给定了问题的初始条件和边界条件，显式公式通常可以直接求解。但是，由于公式的右边含有负系数项，这种显式公式是不稳定的。有关稳定性分析的方法请参阅本书附录 E。根据经验，当所有已知量都位于有限差分公式的右边时，如果其中含有负数系数，则解是不稳定的。这一点用正则定律说明就更明确：

对于

$$u_{i,n+1} = Au_{i+1,n} + Bu_{i,n} + Cu_{i-1,n} \tag{8.59}$$

如果 A、B 和 C 均为正数，且 $A+B+C\leqslant 1$，则该数值计算公式是稳定的。

为了克服不稳定问题，我们用向前差分代换式（8.22）的一阶导数，即

$$\left.\frac{\partial u}{\partial t}\right|_{i,n} = \frac{1}{\Delta t}(u_{i,n+1} - u_{i,n}) + O(\Delta t) \tag{8.60}$$

将式（8.56）和式（8.60）代入式（8.22），可得如下显式公式：

$$u_{i,n+1} = \left(\frac{\alpha\Delta t}{\Delta x^2}\right)u_{i+1,n} + \left(1 - 2\frac{\alpha\Delta t}{\Delta x^2}\right)u_{i,n} + \left(\frac{\alpha\Delta t}{\Delta x^2}\right)u_{i-1,n} + O(\Delta x^2 + \Delta t) \tag{8.61}$$

要获得稳定解，由正则定律的要求可知

$$\left(1 - 2\frac{\alpha\Delta t}{\Delta x^2}\right) \geqslant 0 \tag{8.62}$$

重排该式，得到

$$\frac{\alpha\Delta t}{\Delta x^2} \leqslant \frac{1}{2} \tag{8.63}$$

这个不等式确定了 x 方向的积分步长 Δx 与 t 方向的积分步长 Δt 之间的关系。当 Δx 变小时，Δt 就要变得更小，这样会延长所需的计算时间。

如果在式（8.62）和式（8.63）中取等号，则

$$\frac{\alpha\Delta t}{\Delta x^2} = \frac{1}{2} \tag{8.64}$$

式（8.61）就简化为

$$u_{i,n+1} = \frac{1}{2}(u_{i+1,n} + u_{i-1,n}) + O(\Delta x^2 + \Delta t) \tag{8.65}$$

该式（8.65）由当前时间步长 n 上节点 i 的左右节点的值，计算下一个时间步长（$n+1$）上的节点 i 的因变量值。图 8.8 显示了这个公式的计算方式。

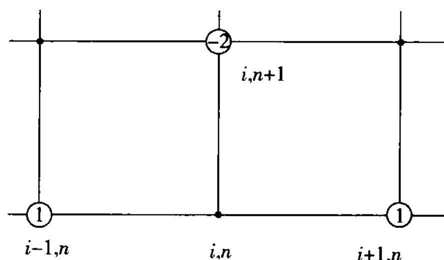


图 8.8　式（8.65）的计算方式

需要强调的是用向前差分代替一阶导数引入的误差为 $O(\Delta t)$ 阶，因此，式（8.61）的误差在时间上为 $O(\Delta t)$ 阶，在 x 上为 $O(\Delta x^2)$ 阶。但是，在这里，如何获得稳定解比提高计算准确度更重要。

如下非齐次抛物型方程

$$\frac{\partial u}{\partial t} = \alpha\frac{\partial^2 u}{\partial x^2} + f(x,t) \tag{8.66}$$

的有限差分解由以下显式公式给出：

$$u_{i,n+1} = \left(\frac{\alpha\Delta t}{\Delta x^2}\right)u_{i+1,n} + \left(1 - 2\frac{\alpha\Delta t}{\Delta x^2}\right)u_{i,n} + \left(\frac{\alpha\Delta t}{\Delta x^2}\right)u_{i-1,n} + (\Delta t)f_{i,n} + O(\Delta x^2 + \Delta t) \tag{8.67}$$

在扩散问题中遇到的方程就是式（8.66）类型的方程。

同理，对于二维抛物型方程

$$\frac{\partial u}{\partial t} = \alpha\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) + f(x,y,t) \tag{8.68}$$

可得

$$u_{i,j,n+1}=\left(\frac{\alpha\Delta t}{\Delta x^2}\right)(u_{i+1,j,n}+u_{i-1,j,n})+\left(\frac{\alpha\Delta t}{\Delta y^2}\right)(u_{i,j+1,n}+u_{i,j-1,n})$$
$$+\left(1-2\frac{\alpha\Delta t}{\Delta x^2}-2\frac{\alpha\Delta t}{\Delta y^2}\right)u_{i,j,n}+(\Delta t)f_{i,j,n}+O(\Delta x^2+\Delta y^2+\Delta t) \tag{8.69}$$

由正则定律可得以下稳定条件：

$$1-2\alpha\Delta t\left(\frac{1}{\Delta x^2}+\frac{1}{\Delta y^2}\right)\geqslant 0 \tag{8.70}$$

重排之后，得到

$$\frac{1}{\Delta x^2}+\frac{1}{\Delta y^2}\leqslant\frac{1}{2\alpha\Delta t} \tag{8.71}$$

通过一些代数运算（Constantinides 和 Mostoufi，1999），该稳定条件可以变换为

$$\frac{\alpha\Delta t}{\Delta x^2+\Delta y^2}\leqslant\frac{1}{8} \tag{8.72}$$

如果在式（8.69）中加入来自$\partial^2u/\partial z^2$的项，就可以导出三维抛物型方程的公式。此时，以上稳定条件不等式的右边是1/18。

在8.4节已经提到，抛物型偏微分方程可以有Dirichlet、Neumann、Cauchy或者Robbins类型的初始条件和边界条件。图8.2显示了扩散问题的初始条件和边界条件的例子。如果边界条件中含有微分，也必须像微分方程那样，用同样的有限差分网格将微分离散化。对于Dirichlet类型的条件，只要把边界上的因变量值设为给定的边界值即可。对于Neumann和Robbins类型的条件，如同8.5.1节所述的那样，边界上因变量的变化需要用有限差分近似值来表示，这样，就增加了新的代数方程，一起组成代数方程组求解。

例8.2 人体白细胞在人造血管材料上的迁移。

问题陈述：

人体白细胞在人造血管材料表面的流动和迁移可以用扩散对流方程来模拟（参见8.2.3节），即

$$\frac{\partial C}{\partial t}=\mu_D\frac{\partial^2 C}{\partial z^2}-v_{eff}\frac{\partial C}{\partial z}$$

请用有限差分近似公式表示偏导数，求该方程的数值解。初始条件和边界条件如下：

初始条件：$t=0$时，对于所有$z>0$有$C=0$

边界条件：$t>0$时，当$z=0$，$C=1\times10^6$；当$z=0.2$，$C=0$。

常数为

随机迁移系数：$\mu_D=1\times10^{-4}\mathrm{cm^2/s}$

细胞定向迁移速度：$v_{eff}=1\times10^{-5}\text{cm}^2/\text{s}$

解：

用向前有限差分表示时间导数，即

$$\left.\frac{\partial C}{\partial t}\right|_{i,n}=\frac{1}{\Delta t}(C_{i,n+1}-C_{i,n})$$

用中心有限差分表示一阶和二阶空间导数，即

$$\left.\frac{\partial C}{\partial z}\right|_{i,n}=\frac{1}{2\Delta z}(C_{i+1,n}-C_{i-1,n})$$

$$\left.\frac{\partial^2 C}{\partial z^2}\right|_{i,n}=\frac{1}{\Delta z^2}(C_{i+1,n}-2C_{i,n}+C_{i-1,n})$$

将这些公式代入方程（8.9），并重排，解得 $C_{i,n+1}$为

$$C_{i,n+1}=\left[\left(\frac{\mu_D\Delta t}{\Delta z^2}\right)-\left(\frac{v_{eff}\Delta t}{2\Delta z}\right)\right]C_{i+1,n}+\left[1-\left(\frac{2\mu_D\Delta t}{\Delta z^2}\right)\right]C_{i,n}+\left[\left(\frac{\mu_D\Delta t}{\Delta z^2}\right)-\left(\frac{v_{eff}\Delta t}{2\Delta z}\right)\right]C_{i-1,n}$$

注意，当 $v_{eff}=0$ 时，这个解与式（8.61）相同。根据正则定律，该解中方括号表示的这些项都必须为正值。要达到这个要求，必须满足以下不等式：

$$\left(\frac{v_{eff}\Delta t}{2\Delta z}\right)<\left(\frac{\mu_D\Delta t}{\Delta z^2}\right)\leqslant\frac{1}{2}$$

本例题有 Dirichlet 类型的边界条件，因此可以直接使用。以下 MATLAB 程序完成问题的求解，并验证了以上不等式是否成立。

```
% example8 _ 2. m-This program calculates and plots the
% concentration profiles of migrating human leukocytes
% on prosthetic materials by solving the parabolic
% partial differential equation using finite differences

clc; clear all;

disp(' Solution of parabolic partial differential equation. ')
disp(' ');
h = input(' Total distance of migration (cm) = ');
tmax = input(' Maximum integration time (s) = ');
nz = input(' Number of divisions in z-direction = ');
nt = input(' Number of divisions in t-direction = ');
mu _ D = input(' Random migration coefficient, mu _ D, (cm^2/s) = ');
v _ eff = input (' Directional migratory velocity, v _ eff, (cm/s)
```

```
=');
disp(' '); disp(' Boundary conditions:'); disp(' ');
b_initial=input(' Density of leukocytes at initial time (cells/cm
^2)=');
b_left=input(' Density of leukocytes at left boundary (cells/cm^
2)=');
b_right=input(' Density of leukocytes at right boundary (cells/cm
^2)=');
% Calculate the increments
dz=h/nz;
z=([0:nz]*dz)';
% Apply stability criteria
if tmax/nt<dz^2/(2*mu_D)
    dt=tmax/nt;
else
    disp(' ')
    disp(' Resetting dt to confirm with stability criteria')
    dt=dz^2/(2*mu_D)
end
if (v_eff*dt)/(2*dz)>=(mu_D*dt)/(dz^2)
    error(' Stability criterion violated')
end

nt=ceil(tmax/dt);
t=[0:nt]*dt;
% Create the solution matrix
C=zeros(nz+1,nt+1);
% Enter the initial conditions into the solution matrix
C(:,1)=b_initial;
% Enter the boundary conditions into the solution matrix
C(1,2:nt+1)=b_left;
% Enter the boundary conditions into the solution matrix
C(nz+1,2:nt+1)=b_right;
% Iteration on t
for n=2:nt
```

```
% Calculate the concentration profile
    for i =2:nz
        A = (mu_D * dt)/(dz^2)-(v_eff * dt)/(2 * dz);
        B =1 -2 * (mu_D * dt)/(dz^2);
        C(i,n +1) =A * C(i +1,n) +B * C(i,n) +A * C(i -1,n);
    end
end
% Plot k concentration profiles
k =5;
new_C(:,1) =C(:,1)/b_left;
count =fix((length(t) -1)/k);
for i =1:k
    new_C(:,i +1) =C(:,1 +i * count)/b_left;
end
clf; figure(1); plot(z, new_C)
xlabel('Distance, cm'); ylabel('Normalized cell density, C/C_0')
title('Normalized Cell Density Profiles')
% Create textarrow
annotation1 =annotation(figure(1),'textarrow',...
    [0.2967 0.4695],[0.2643 0.4713],'String',{'Time'});
```

输入

```
>> example8_2
 Solution of parabolic partial differential equation.

 Total distance of migration (cm) =0.2
 Maximum integration time (s) =40


 Number of divisions in z -direction =100
 Number of divisions in t -direction =100
 Random migration coefficient, mu_D, (cm^2/s) =1e -4
 Directional migratory velocity, v_eff, (cm/s) =1e -5

 Boundary conditions:

 Density of leukocytes at initial time (cells/cm^2) =0
```

```
 Density of leukocytes at left boundary (cells/cm^2) = 1e6
 Density of leukocytes at right boundary (cells/cm^2) = 0

 Resetting dt to confirm with stability criteria
dt =
    0.0200
 >>
```

结果讨论：

图 8.9 是归一化的细胞浓度 C/C_0 随迁移距离变化的曲线。开始时（即 $t=0$ 时），人造血管表面的细胞浓度为 0，左边界上（即 $z=0$ 处）有高浓度的细胞（1×10^6 细胞/cm^2）。然后，如同微分方程的解所表明的，细胞向右边迁移。右边界的边界条件为 Dirichlet 类型，值为 0，这相当于此处细胞浓度始终为 0，也就如同在右边界存在一个无限深的穴，会吸走细胞一样，使得细胞不会在右边界聚集。

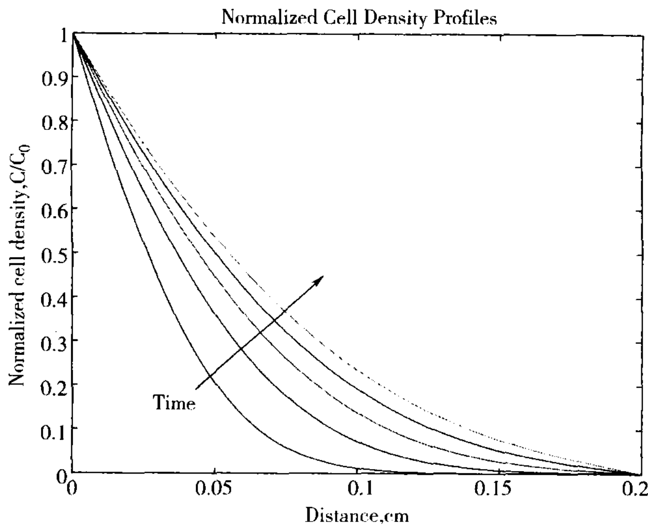


图 8.9　归一化的细胞浓度变化曲线

建议读者用 $v_{eff}=0$ 重新求解此方程，并比较结果的差别。也可以做一下习题 8.4，用 Neumann 边界条件求解这个模型。

隐式方法：下面来看一些求解抛物型方程的隐式方法。如图 8.10 所示的网格显示了 t 轴上的半节点（i, $n+½$），利用这个网格可以建立中心差分公式，提高算法的准确度。与显式公式中用（i,n）节点上的向前差分表示 $\partial u/\partial t$ 不同，这里用半节点上的中心差分表示偏导数，即

$$\left.\frac{\partial u}{\partial t}\right|_{i,n+½} = \frac{1}{\Delta t}(u_{i,n+1} - u_{i,n}) \tag{8.73}$$

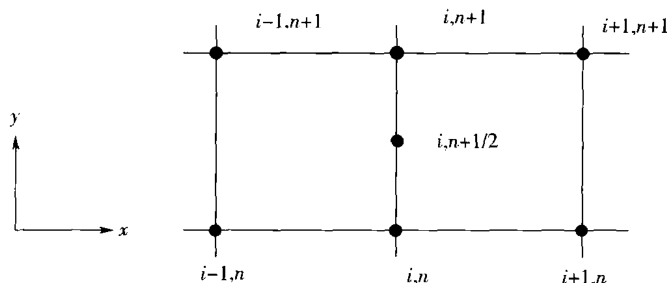


图 8.10 隐式导数公式的有限差分网格

另外，在半节点上的二阶偏导数表示为节点 $(i,n+1)$ 和节点(i,n) 上的中心差分的加权平均，即

$$\begin{aligned}\left.\frac{\partial^2 u}{\partial x^2}\right|_{i,n+½} &= \theta \left.\frac{\partial^2 u}{\partial x^2}\right|_{i,n+1} + (1-\theta)\left.\frac{\partial^2 u}{\partial x^2}\right|_{i,n} \\ &= \theta\left[\frac{1}{\Delta x^2}(u_{i+1,n+1} - 2u_{i,n+1} + u_{i-1,n+1})\right] + \\ &\quad (1-\theta)\left[\frac{1}{\Delta x^2}(u_{i+1,n} - 2u_{i,n} + u_{i-1,n})\right]\end{aligned} \tag{8.74}$$

其中，θ 的取值范围为 $0\leqslant\theta\leqslant1$。将式（8.73）和式（8.74）代入如下方程，即式（8.22）

$$\frac{\partial u}{\partial t} = \alpha\frac{\partial^2 u}{\partial x^2} \tag{8.22}$$

可得抛物型偏微分方程的可变权重隐式近似公式为

$$\begin{aligned}&\alpha\theta\left[\frac{1}{\Delta x^2}(u_{i+1,n+1} - 2u_{i,n+1} + u_{i-1,n+1})\right] - \frac{1}{\Delta t}u_{i,n+1} \\ &= -\alpha(1-\theta)\left[\frac{1}{\Delta x^2}(u_{i+1,n} - 2u_{i,n} + u_{i-1,n})\right] - \frac{1}{\Delta t}u_{i,n}\end{aligned} \tag{8.75}$$

式（8.75）为隐式，因为公式左边包含了不止一个差分网格 $(n+1)$ 节点上的值，也就是时间域上的每一步计算都有多个未知数。

当 $\theta=0$ 时，式（8.75）变成了显式公式（8.67）；当 $\theta=1$ 时，式（8.75）成为

$$-\left(\frac{\alpha\Delta t}{\Delta x^2}\right)u_{i-1,n+1} + \left(1+2\frac{\alpha\Delta t}{\Delta x^2}\right)u_{i,n+1} - \left(\frac{\alpha\Delta t}{\Delta x^2}\right)u_{i+1,n+1} = u_{i,n} \tag{8.76}$$

式（8.76）称为向后隐式近似公式。如果直接用 $(i,n+1)$ 节点上的向后差分近似一阶偏导数，用 $(i,n+1)$ 节点上的中心差分近似二阶偏导数，也可以得到

式（8.76）。

最后，当 $\theta = ½$时，式（8.75）就变成了如下被广泛应用的克拉克-尼科尔森隐式公式

$$
\begin{aligned}
& -\left(\frac{\alpha \Delta t}{\Delta x^2}\right) u_{i-1,n+1} + 2\left(1 + \frac{\alpha \Delta t}{\Delta x^2}\right) u_{i,n+1} - \left(\frac{\alpha \Delta t}{\Delta x^2}\right) u_{i+1,n+1} \\
& = \left(\frac{\alpha \Delta t}{\Delta x^2}\right) u_{i-1,n} + 2\left(1 - \frac{\alpha \Delta t}{\Delta x^2}\right) u_{i,n} + \left(\frac{\alpha \Delta t}{\Delta x^2}\right) u_{i+1,n}
\end{aligned} \tag{8.77}
$$

为了使用这种隐式算法求解以下非齐次抛物型方程

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2} + f(x,t) \tag{8.66}$$

还需要计算中间节点（i, $n + ½$）上的f函数值，可以取节点（$i,n + 1$）和(i,n)上f值的平均，即

$$f_{i,n+½} = \frac{1}{2}(f_{i,n+1} + f_{i,n}) \tag{8.78}$$

将式（8.73）、$\theta = ½$时的式（8.74）以及式（8.78）同时代入式（8.66），可得

$$
\begin{aligned}
& -\left(\frac{\alpha \Delta t}{\Delta x^2}\right) u_{i-1,n+1} + 2\left(1 + \frac{\alpha \Delta t}{\Delta x^2}\right) u_{i,n+1} - \left(\frac{\alpha \Delta t}{\Delta x^2}\right) u_{i+1,n+1} - (\Delta t) f_{i,n+1} \\
& = \left(\frac{\alpha \Delta t}{\Delta x^2}\right) u_{i-1,n} + 2\left(1 - \frac{\alpha \Delta t}{\Delta x^2}\right) u_{i,n} + \left(\frac{\alpha \Delta t}{\Delta x^2}\right) u_{i+1,n} + (\Delta t) f_{i,n}
\end{aligned} \tag{8.79}
$$

式（8.79）就是式（8.66）非齐次抛物型偏微分方程的克拉克-尼科尔森隐式公式。

对整个差分网格的每个节点写出隐式公式，就得到一个线性代数方程组，其系数矩阵通常是一个三对角矩阵。这种方程组可以用第 4 章介绍的高斯消元法求解，或者用更有效的 Thomas 算法（Lapidus 和 Pinder，1982）求解，Thomas 算法是高斯消元法的一种变异。

上述隐式公式是绝对稳定的。一般而言，大部分显式有限差分近似公式是条件稳定的，而大部分隐式近似公式则是绝对稳定的。但是，显式方法比隐式方法容易进行求解计算。

8.5.3 双曲型偏微分方程

与振动过程相关的物理问题会产生包含二阶偏微分方程的双曲型数学模型。例如，一维波动方程为

$$\rho \frac{\partial^2 u}{\partial t^2} = T_0 \frac{\partial^2 u}{\partial x^2} + f(x,t) \tag{8.80}$$

该方程描述了在张力 T_0 和外力$f(x,t)$ 作用下，振动弦的横向运动。当密度 ρ 为

常数时，该方程可以写为

$$\frac{\partial^2 u}{\partial t^2} = a^2 \frac{\partial^2 u}{\partial x^2} + F(x,t) \tag{8.81}$$

其中，$a^2 = \frac{T_0}{\rho}, F(x,t) = \frac{1}{\rho} f(x,t)$。

如果弦上没有施加外力，式（8.81）就成了如下齐次方程：

$$\frac{\partial^2 u}{\partial t^2} = a^2 \frac{\partial^2 u}{\partial x^2} \tag{8.82}$$

为了求取该齐次方程的数值解，用中心有限差分展开其中的两个二阶偏导数，可得

$$\frac{u_{i,n+1} - 2u_{i,n} + u_{i,n-1}}{\Delta t^2} = a^2 \left(\frac{u_{i+1,n} - 2u_{i,n} + u_{i-1,n}}{\Delta x^2} \right) + O(\Delta x^2 + \Delta t^2) \tag{8.83}$$

重排该方程，求得 $u_{i,n+1}$ 为

$$u_{i,n+1} = 2\left(1 - \frac{a^2 \Delta t^2}{\Delta x^2}\right) u_{i,n} + \frac{a^2 \Delta t^2}{\Delta x^2}(u_{i+1,n} + u_{i-1,n}) - u_{i,n-1} + O(\Delta x^2 + \Delta t^2) \tag{8.84}$$

这就是双曲型式（8.82）的显式数值求解法。

将式（8.59）的正则定律用于该方程，可知，当以下不等式

$$\frac{a^2 \Delta t^2}{\Delta x^2} \leqslant 1 \tag{8.85}$$

成立时，式（8.84）的解是稳定的。

同理，二维双曲型方程的齐次形式为

$$\frac{\partial^2 u}{\partial t^2} = a^2 \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) \tag{8.86}$$

用中心有限差分近似公式展开，可得

$$\begin{aligned} \frac{u_{i,j,n+1} - 2u_{i,j,n} + u_{i,j,n-1}}{\Delta t^2} &= a^2 \left(\frac{u_{i+1,j,n} - 2u_{i,j,n} + u_{i-1,j,n}}{\Delta x^2} \right) \\ &\quad + a^2 \left(\frac{u_{i,j+1,n} - 2u_{i,j,n} + u_{i,j-1,n}}{\Delta y^2} \right) + O(\Delta x^2 + \Delta y^2 + \Delta t^2) \end{aligned} \tag{8.87}$$

重排式（8.87），并设 x 轴和 y 轴的网格等距，可得如下显式公式：

$$\begin{aligned} u_{i,j,n+1} &= 2\left[1 - 2\left(\frac{a^2 \Delta t^2}{\Delta x^2}\right)\right] u_{i,j,n} - u_{i,j,n-1} \\ &\quad + \frac{a^2 \Delta t^2}{\Delta x^2}(u_{i+1,j,n} + u_{i-1,j,n} + u_{i,j+1,n} + u_{i,j-1,n}) \end{aligned} \tag{8.88}$$

当不等式

$$\frac{a^2 \Delta t^2}{\Delta x^2} \leqslant \frac{1}{2} \tag{8.89}$$

成立时，解是稳定的。

用可变权重方法可以导出双曲型偏微分方程的隐式求解公式，其中的空间偏导数用 $(n+1)$、n 和 $(n-1)$ 节点上差分值的加权平均来表示，可得方程 (8.82) 的隐式求解公式如下：

$$\begin{aligned}\frac{u_{i,n+1} - 2u_{i,n} + u_{i,n-1}}{\Delta t^2} &= \frac{a^2}{\Delta x^2}[\theta(u_{i+1,n+1} - 2u_{i,n+1} + u_{i-1,n+1}) \\ &+ (1-2\theta)(u_{i+1,n} - 2u_{i,n} + u_{i-1,n}) + \theta(u_{i+1,n-1} - 2u_{i,n-1} + u_{i-1,n-1})]\end{aligned} \tag{8.90}$$

其中，$0 \leqslant \theta \leqslant 1$。当 $\theta = 0$ 时，式（8.90）就是显式公式（8.83）；当 $\theta = 1/2$ 时，式（8.90）是克拉克-尼科尔森隐式公式。隐式算法产生三对角线性代数方程组，可以用第 4 章介绍的高斯消元法求解。

8.6 极坐标系统

本章以上推导的所有方法都是基于笛卡尔坐标系统。但是，用偏微分方程仿真的对象经常具有圆形、圆柱形或球形的特征，因此，可以对有限差分近似公式进行修正，以便于处理具有这类几何特性的问题。

圆柱形对象用极坐标表示比较方便。如图 8.11 所示，利用如下变换关系式可以从笛卡尔坐标系统转变到极坐标系统：

$$\begin{aligned} x &= r\cos\theta \qquad & y &= r\sin\theta \\ r &= \sqrt{x^2 + y^2} \qquad & \theta &= \arctan\frac{y}{x} \end{aligned} \tag{8.91}$$

这样，极坐标系统下的拉普拉斯算子就变为

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \frac{\partial^2 u}{\partial r^2} + \frac{1}{r}\frac{\partial u}{\partial r} + \frac{1}{r^2}\frac{\partial^2 u}{\partial \theta^2} \tag{8.92}$$

极坐标系统下的 Fick 第二扩散定律则为

$$\frac{\partial C}{\partial t} = D\left(\frac{\partial^2 C}{\partial r^2} + \frac{1}{r}\frac{\partial C}{\partial r} + \frac{1}{r^2}\frac{\partial^2 C}{\partial \theta^2} + \frac{\partial^2 C}{\partial z^2}\right) \tag{8.93}$$

利用图 8.12 所示的极坐标有限差分网格，可得如下偏导数的近似公式：

$$\left.\frac{\partial^2 u}{\partial r^2}\right|_{i,j} = \frac{1}{\Delta r^2}(u_{i,j+1} - 2u_{i,j} + u_{i,j-1}) \tag{8.94}$$

$$\left.\frac{\partial^2 u}{\partial \theta^2}\right|_{i,j} = \frac{1}{\Delta \theta^2}(u_{i+1,j} - 2u_{i,j} + u_{i-1,j}) \tag{8.95}$$

$$\left.\frac{\partial u}{\partial r}\right|_{i,j} = \frac{1}{2\Delta r}(u_{i,j+1} - u_{i,j-1}) \tag{8.96}$$

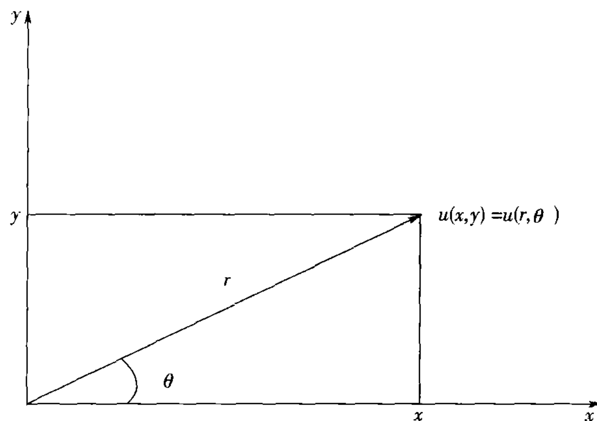


图 8.11　笛卡尔坐标到极坐标的转换

式中　i，j——θ 轴和 r 轴的计数器。

同样，通过增加下标，可以表示图 8.12 没有显示的 z 轴和 t 时间轴上的偏导数。

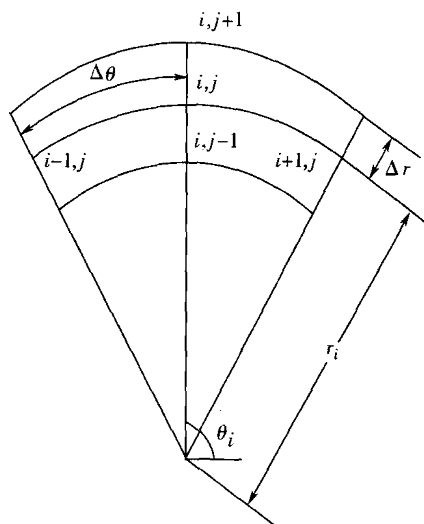


图 8.12　极坐标的有限差分网格

8.7　稳定性分析

第 7 章讲述了用数值积分方法求解常微分方程的稳定性问题，在偏微分方程

的数值求解中同样必须重视稳定性问题。本书附录 E 用著名的冯·诺依曼方法讨论了有限差分近似公式的稳定性问题。

8.8 MATLAB 中的偏微分方程工具箱

MATLAB 有一个功能很强的求解线性和非线性偏微分方程的工具箱，简称 PDE 工具箱。但是，这个工具箱需要单独购买，并不是所有 MATLAB 用户都有的。PDE 工具箱利用有限元法求解二维空间的偏微分方程，工具箱的基本方程为

$$-\nabla\cdot(c\nabla u)+au=f \tag{8.97}$$

式中 c，a，f——解域中的复数函数，也可以是 u 的函数。

PDE 工具箱还可以求解方程

$$d\frac{\partial u}{\partial t}-\nabla\cdot(c\nabla u)+au=f \tag{8.98}$$

以及方程

$$d\frac{\partial^2 u}{\partial t^2}-\nabla\cdot(c\nabla u)+au=f \tag{8.99}$$

式中 d，c，a，f——解域中的常数或者复数函数，也可以是时间函数；

符号 ∇——矢量微分算子，也就是梯度。

注意，不要把它与向后差分算子 ∇相混淆。

不论式（8.97）、式（8.98）和式（8.99）具有怎样的系数和边界条件，PDE 工具箱都分别把这 3 个方程称为椭圆型、抛物型和双曲型方程。

使用 PDE 工具箱求解偏微分方程时，用 pdetool 指令可以直接进入 PDE 工具箱的图形用户界面。在这个独立的环境下，用户可以定义二维几何形状，加入边界条件，求解偏微分方程，并作图显示求解结果。下面的例 8.3 将用一维形式[即式（8.24）] Fick 第二扩散定律的求解过程说明 PDE 工具箱的用法。

例 8.3 利用 PDE 工具箱求解 Fick 第二扩散定律

问题陈述：

分子的简单跨膜扩散可以用一维形式的 Fick 第二定律来模拟

$$\frac{\partial C_A}{\partial t}=D_{AB}\frac{\partial^2 C_A}{\partial x^2} \tag{8.24}$$

式中 C_A——A 成分在膜中的浓度；

D_{AB}——A 成分在膜中的扩散系数；

x——扩散距离。

这是一个一维非稳态抛物型偏微分方程，初始条件和边界条件如下。请用 PDE 工具箱求解该方程。

初始条件：当 $t = 0$ 时，对于所有 $x > 0$，有 $C = 0$；

边界条件：当 $t > 0$ 时，若 $x = 0$，则 $C = 1$；若 $x = 0.2$，则 $C = 0$。

扩散系数常数的值为 $D_{AB} = 1 \times 10^{-4} \mathrm{cm}^2/\mathrm{s}$。

解：

在 MATLAB 指令窗中键入指令：

```
>> pdetool
```

出现如图 8.13 所示的一个空白 PDE 工具箱窗口。

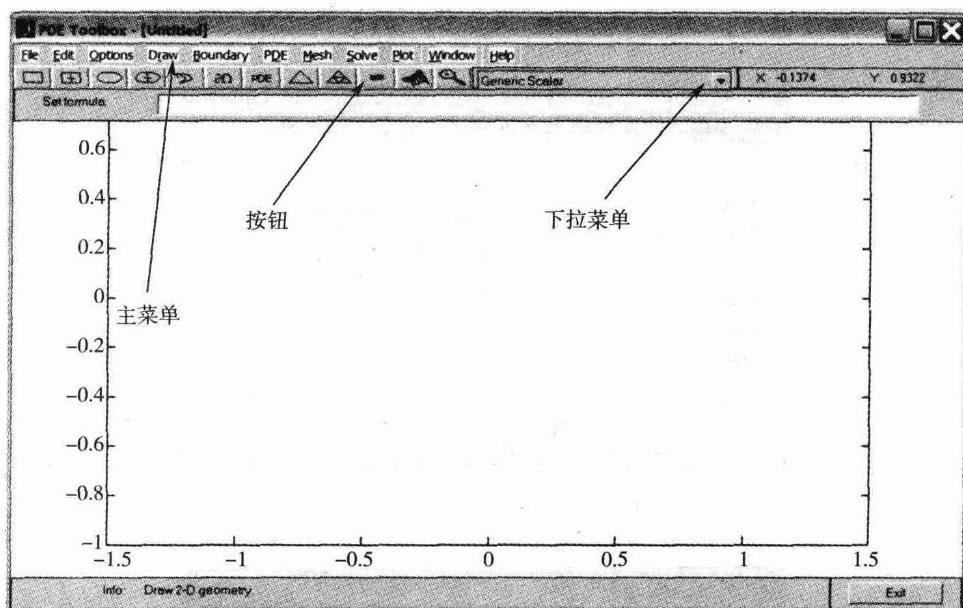


图 8.13　PDE 工具箱窗口及其菜单

请注意窗口中的主菜单、按钮栏以及下拉菜单。从下拉菜单中选取“Diffusion”（扩散）项，并点击“PDE”按钮，如图 8.14 所示，进入参数设置窗，选取抛物型方程，并输入系数的值。

然后，选择按钮栏最左边的矩形按钮，如图 8.15 所示，在窗口中拉出一个小矩形。双击该矩形，出现如图 8.16 所示的对话框，在对象对话框中设定坐标值。从“Options”菜单中选择“Axes Limits…”一项，如图 8.17 所示，设置 X 轴和 Y 轴的取值范围。

单击图 8.15 中的∂Ω 按钮，矩形的边界线变成红色，此时就可以设定边界条件。如图 8.18 ~ 图 8.21 所示，双击矩形的每一条边分别设置边界条件。

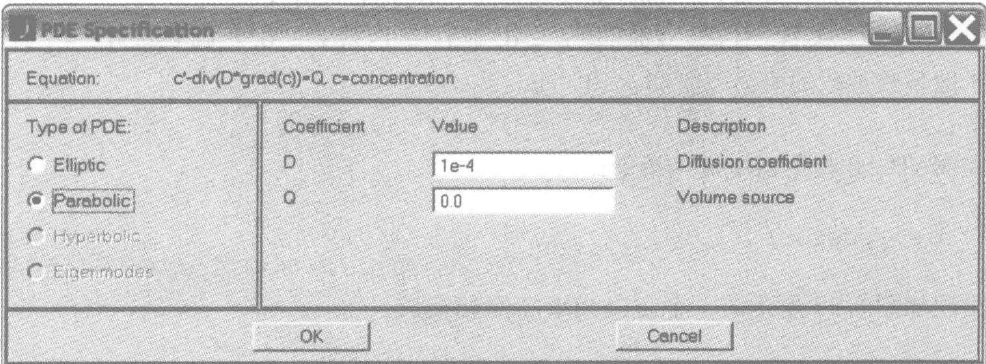


图 8.14 PDE 参数设置对话框

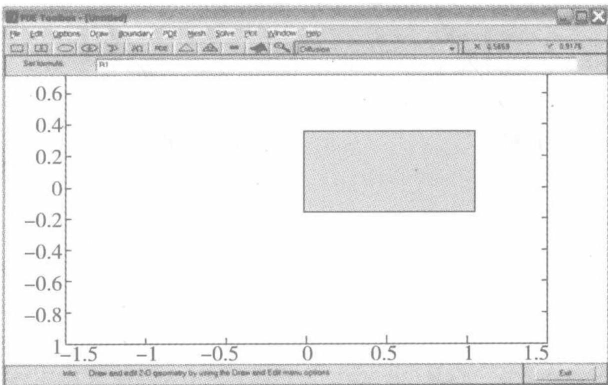


图 8.15 待求解问题的几何图形

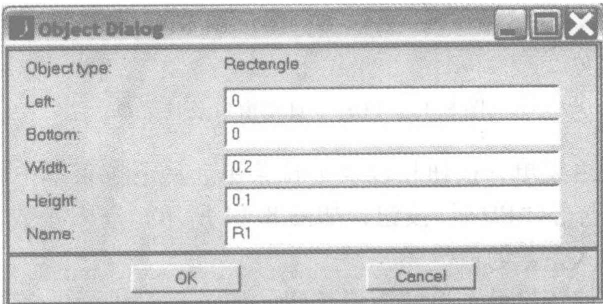


图 8.16 对象对话框（设置坐标值，确定对象的大小）

选择“Solve”菜单的“Parameters”项，如图 8.22 所示，设置积分时间和初始条件。

最后，单击“=”按钮求解微分方程，可得如图 8.23 所示的输出结果，图中显示了膜对象中自左向右的扩散过程。

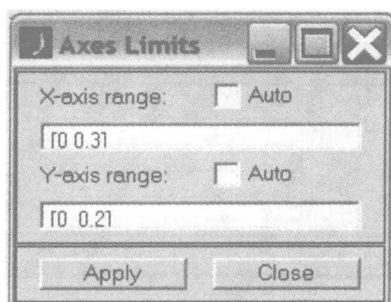


图 8.17　设定坐标轴取值范围的对话框

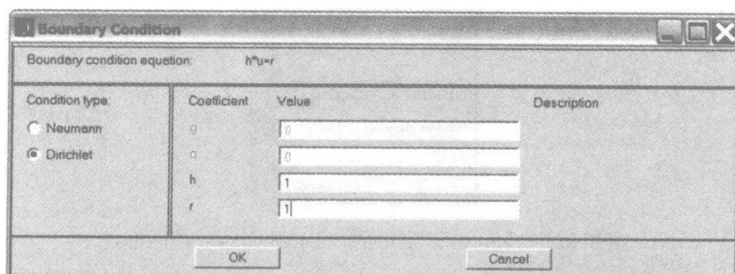


图 8.18　边界条件对话框：设置左边界

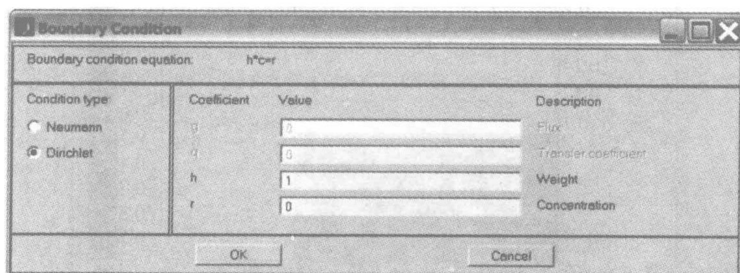


图 8.19　边界条件对话框：设置右边界

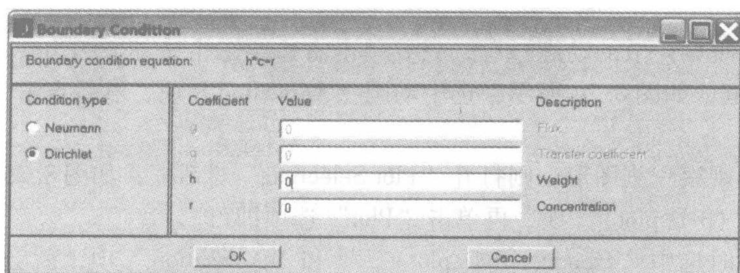


图 8.20　边界条件对话框：设置下边界

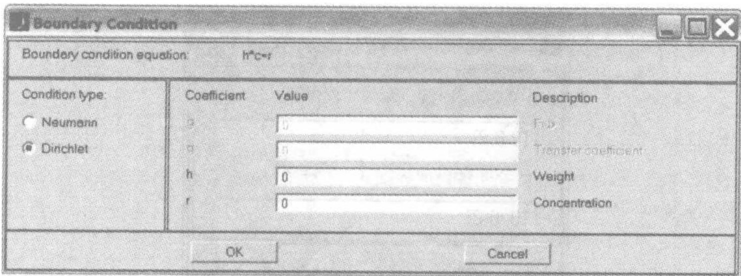


图 8.21 边界条件对话框：设置上边界

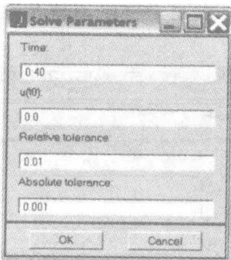


图 8.22 “Solve” 菜单的 “Parameters” 对话框

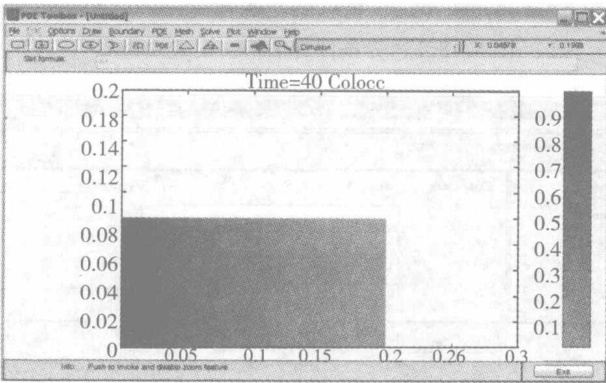


图 8.23 微分方程的解：扩散的求解结果

如果要观察扩散的动态过程，只要单击右边第二个按钮，打开 “Plot Selection” 对话框，如图 8.24 所示，选中动画 “Animation” 项，然后点击 “Plot” 选项即可。

如果要观察三维图形，则打开 “Plot Selection” 对话框，如图 8.25 所示，选中 “Height (3-D plot)” 项，再单击 “Plot” 选项即可。

如果同时选中 “Height (3-D plot)” 项和 “Animation” 项，就会产生一个扩散过程的三维动画显示。建议读者自己测试一下其他作图功能，并增加积分时

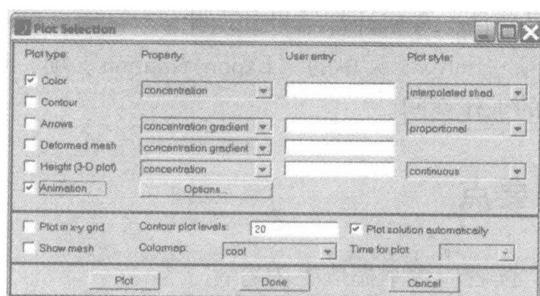
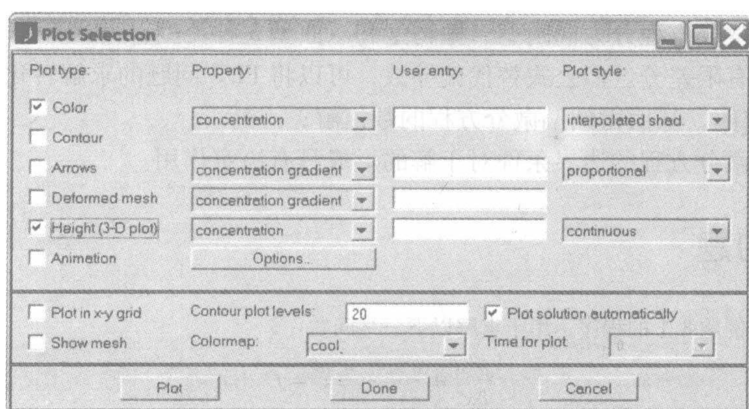


图 8.24　“Plot Selection”对话框



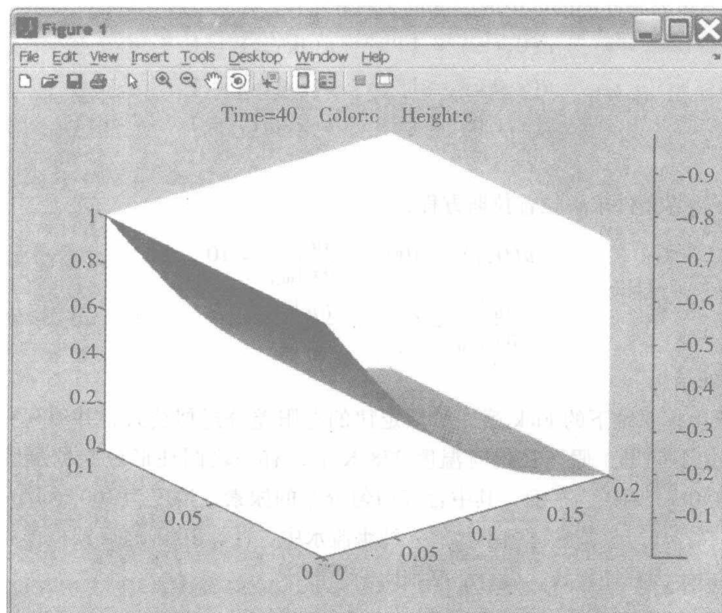


图 8.25　“Plot Selection”对话框的设置以及三维图形输出结果

间、或者用其他边界条件，考察计算结果的变化。

利用 PDE 工具箱“Solve”菜单的“Export Solution”项，可以将数值解的结果输送到 MATLAB 的工作空间。

8.9 本章学习要点

学完本章之后，读者应该掌握以下内容：

1）用偏微分方程可以模拟具有多个自变量的生物医学系统的动态变化过程；

2）偏微分方程分为椭圆型、抛物型和双曲型 3 种类型；

3）用有限差分近似公式替代偏导数，可以将 PDE 问题的求解转化成为代数方程组的求解，从而得到偏微分方程的数值解；

4）偏微分方程的边界条件对于解的形成具有决定作用。

8.10 习题

8.1 请修改例题 8.1 的程序，用于求解以下三维问题：

$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2}=f$$

并计算具有如下边界条件的固体中的因变量 u 的分布：

$$u(0,y,z)=1 \qquad u(1,y,z)=1$$
$$u(x,0,z)=1 \qquad u(x,1,z)=1$$
$$u(x,y,0)=1 \qquad u(x,y,1)=1$$

假设 $f=5$。

8.2 请用如下边界条件求解拉普拉斯方程：

$$u(0,y)=100 \qquad \left.\frac{\partial u}{\partial x}\right|_{10,y}=10$$
$$\left.\frac{\partial u}{\partial y}\right|_{x,0}=0 \qquad \left.\frac{\partial u}{\partial y}\right|_{x,1}=0$$

并解释所得到的结果。

8.3 请推导极坐标系统下的 Fick 第二扩散定律的有限差分近似公式，并编写一个 MATLAB 程序用于求解如下问题：假设在绝对温度 278 K 下，有一段圆柱形琼脂糖凝胶体，直径为 0.03m，长为 1.0m，两端为平面，其中含有均匀分布的尿素，浓度为 100 mol/m^3，扩散系数为 5×10^{-10}m^2/s。如果该圆柱体突然被浸入纯水流水中，计算 100h 之后圆柱体中心点的尿素浓度。提示：由于圆柱体很长，可以忽略轴向的扩散。

8.4 例 8.2 已经用如下扩散对流方程求解了人体白细胞在人造血管材料表面的流动和迁移过程：

$$\frac{\partial C}{\partial t}=\mu_D\frac{\partial^2 C}{\partial z^2}-v_{eff}\frac{\partial C}{\partial z}$$

假设细胞定向迁移的速度可以忽略不计，即 $v_{eff}=0.0\text{cm/s}$，并且随机迁移系数为 $\mu_D=1\times10^{-4}$ cm^2/s。当右边界具有如下 Neumann 边界条件时，请求解该方程。

初始条件：当 $t=0$ 时，对于所有 $z>0$，有 $C=0$；

边界条件：当 $t>0$ 时，若 $z=0$，则 $C=1\times10^6$；若 $z=0.2$，则 $\frac{\partial C}{\partial z}=0$。

方程的积分时间取 100s，并将求解结果与例题 8.2 的结果进行比较。

8.5　在皮肤表面经皮给药的过程中，药物分子穿过皮肤的运动机制可以用 Fick 第二扩散定律等扩散方程来模拟（Kubota 等人，2002；Simon 和 Loney，2003）。图 8.26 显示了这个过程，药物首先要通过扩散穿过厚度为 L_m 的给药基质，药物在给药基质中的浓度 C_m 可以用如下方程描述：

$$\frac{\partial C_m}{\partial t}=D_m\frac{\partial^2 C_m}{\partial x^2}\quad -L_m\leqslant x\leqslant 0,\quad t>0$$

接着，药物进入并穿过厚度为 L_s 的皮肤层，药物在皮肤层中的浓度 C_s 用如下方程描述：

$$\frac{\partial C_s}{\partial t}=D_s\frac{\partial^2 C_s}{\partial x^2}\quad 0\leqslant x\leqslant L_s;\quad t>0$$

式中　D_m——给药基质中药物的有效扩散系数；

　　　D_s——皮肤中药物的扩散系数。

这两个方程在给药基质与皮肤的接触界面上衔接在一起。最后，药物被体内的受体细胞吸收。

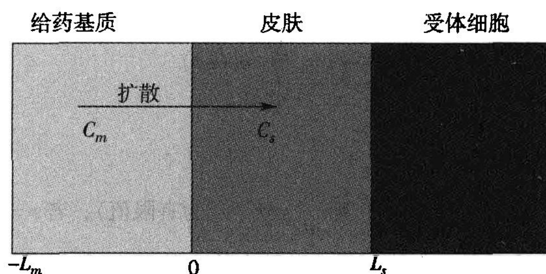


图 8.26　经皮给药的药物扩散过程

请联立求解以上两个方程。初始条件和边界条件如下：

初始条件：

当 $t=0$ 时　$C_m(x,0)=C_{m0}\qquad -L_m\leqslant x<0$

　　　　　　$C_s(x,0)=0\qquad 0\leqslant x\leqslant L_s$

边界条件：

当 $x=-L_m$ 时　$\frac{\partial C_m(-L_m,t)}{\partial x}=0\qquad 0\leqslant t\leqslant T$

当 $x=0$ 时　$-D_m\frac{\partial C_m(0,t)}{\partial x}=-D_s\frac{\partial C_s(0,t)}{\partial x}\qquad 0\leqslant t\leqslant T$

当 $x=0$ 时　$K_m C_m(0,t)=C_s(0,t)$　$0\leqslant t\leqslant T$

当 $x=L_s$ 时　$-D_s\dfrac{\partial C_s(L_s,t)}{\partial x}=K_{cl}C_s(L_s,t)$　$0<t\leqslant T$

式中　K_m——给药基质与皮肤接触界面上的隔离系数；

K_{cl}——皮肤与受体界面上单位面积上单位药物浓度的清除率。

各个常数的值设定如下：

$L_m=0.004\text{cm}$　$L_s=0.025\text{cm}$　$D_m=9.72\times10^{-6}\text{cm}^2/\text{h}$

$D_s=11.25\times10^{-6}\text{cm}^2/\text{h}$　$K_m=0.5$　$K_{cl}=25\text{cm/h}$

$C_{m0}=0.2\text{g/cm}^3$　$T=6\text{h}$

8.6　动脉血管中的血流可以看作刚性圆管中不可压缩牛顿流体的层流。血管半径为 R，并且受到周期性变化的压力场的作用。假设只存在轴向流动，即 $u_r=u_\theta=0$，且轴向流速 u_z 只是半径位置的函数。可以用柱形坐标系统下的纳维-斯托克斯（Navier-Stokes）方程模拟流速 u_z（Truskey 等人，2004），即

$$\frac{\partial u_z}{\partial t}=v\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial u_z}{\partial r}\right)-\frac{1}{\rho}\frac{\partial p}{\partial z}\tag{1}$$

式中　v——动态粘滞度；

ρ——流体的密度；

r——血管半径方向的位置。

设压力梯度周期性变化的频率为 ω 弧度/秒（rad/s），用以模拟心脏的泵血活动，其方程为

$$-\frac{\partial p}{\partial z}=\frac{\Delta p}{L}\cos(\omega t)\tag{2}$$

初始条件和边界条件如下：

初始条件：　当 $t=0$ 时，$u_z=u_{z0}$

边界条件：　当 $t>0$ 时，若 $r=0$，则 $\dfrac{\partial u_z}{\partial r}=0$（$u_z$ 为有限值）；若 $r=R$，则 $u_z=0$。

以下是描述人体左心室主动脉、人体血流、以及人体心脏泵血活动特性的有关参数，可以用这些常数解题：

$R=0.425\text{cm}$　$v=0.09\text{cm}^2/\text{s}$　$\rho=1\text{g/cm}^3$

$\omega=3$ 周/秒 $=6\pi$ rad/s

$\dfrac{\Delta p}{L}=1\ \text{dyne/cm}^3=1(\text{g}\cdot\text{cm/s}^2)/\text{cm}^3$

由这些常数可以求得 Womersley 数的值为

$$\alpha=R\sqrt{\frac{\omega}{v}}=6.15$$

该值与文献报道的人体左心室主动脉数据一致（参见 Truskey 等人，2004）。

8.7　利用激光切除生物组织是外科手术治疗的常用方法。当组织受到激光照射时，光子会穿入靶点，将能量带给组织。其中的一部分能量被组织吸收，并转化为热能，激光的作用就是

提供热源，可以使生物组织的温度上升。但是，温度不可能无限制地一直上升。到达某个临界温度以后，组织中的水就会大量汽化，导致气泡的形成，然后在更高温度的作用下发生组织热解。这一系列汽化、气泡形成、以及高温分解的过程就形成了激光切除的整个过程。

对于一维半无限纯吸收介质，Enderle 等人（2005）建立了如下方程，用于模拟切除之前加热阶段组织的温度变化：

$$\frac{\partial(\rho cT)}{\partial t} = k\frac{\partial^2 T}{\partial x^2} + \mu_a I_0 e^{-\mu_a x}$$

式中　T——组织的温度；

ρ——密度；

c——热容量；

μ_a——吸收系数；

k——组织的热传导率；

I_0——激光束的强度。

请求解该方程，并作图显示温度随组织深度和时间的变化过程，确定组织温度达到 100℃（即切除温度）的时间。初始条件和边界条件如下：

初始条件：当 $t = 0$ 时，对于所有 x，有 $T = 37℃$

边界条件：当 $t > 0$ 时，若 $x = 0$，则 $-k\dfrac{\partial T}{\partial x} = \dfrac{q}{A}$

若 $x = 3.0\text{mm}$，则 $\dfrac{\partial T}{\partial x} = 0$

本题用到的常数值如下：

$\rho = 1\text{g/cm}^3$　　$c = 3.14\text{J/(g}\cdot\text{K)}$　　$\mu_a = 0.25\text{cm}^{-1}$

$k = 0.0054\text{W/(cm}\cdot\text{K)}$　　$\dfrac{q}{A} = 5\text{J/s/cm}^2$　　$I_0 = 100\text{W/cm}^2$

8.11 参考文献

Constantinides, A., and Mostoufi, N. 1999. *Numerical Methods for Chemical Engineers with MATLAB Applications*. Upper Saddle River, NJ: Prentice Hall PTR.

Enderle, J. D., Blanchard, S. M., and Bronzino, J. D. 2005. *Introduction to Biomedical Engineering*. 2nd ed. San Diego, CA: Academic Press.

Kubota, K., Dey, F., Matar, S. A., and Twizell, E. H. 2002. A repeated dose model of percutaneous drug absorption. *Appl. Math Modelling*, **26**, 529-544.

Lapidus, L., and Pinder, G. F. 1982. *Numerical Solution of Partial Differential Equations in Science and Engineering*, New York: Wiley.

Randomsky, M. L., Whaley, K. J., Cone, R. A., and Saltzman, W. M. 1990. Macromolecules released from polymers: diffusion into unstirred fluids. *Biomaterials*, **11**, 619-624.

Rosenson-Schloss, R. S., Chang, C. C., Constantinides, A., Moghe, P. V. 2002. Alteration of Leukocyte Migration on Prosthetic Material, ePTFE, Under Flow: Role of CD43, an Adhesion Regulator Molecule. *Journal of Biomed. Mater. Res.*, **60**, 8-19.

Simon, L. and Loney, N. W. 2003. An analytical solution for percutaneous drug absorption. *Proceedings of IEEE 29th Annual Bioengineering Conference*, 22-23 March, 2003, pp. 303-304.

Truskey, G. A., Yuan, F., and Katz, D. F. 2004. *Transport Phenomena in Biological Systems*. Upper Saddle River, NJ: Pearson Prentice Hall.

第4部分　建模工具及其应用

第9章　测量、建模与统计分析

9.1　数值方法的作用

数值方法远不只是“数字处理”而已，它是从数据中提取信息的方法。数值方法指导我们如何组织并解释数据。例如，生物医学工程人员经常应用数值方法挖掘和搜索生物分子序列中的规律和线索，并确定这些规律的正确程度以及是否具有普遍意义。

统计学是数学的一个分支，用于分析测量误差和生物的变异性，并描述测量数据的特性，从观察和测量结果中归纳出结论。因此，生物医学领域的实验设计、测量方法选择、以及求解结果的解释等都要用到统计模型。

本章围绕寻找（即检测）和解释（即评价）这两个任务来介绍统计学的方法。由于生物过程本身的机制以及数据采集设备“噪声”等因素的影响，任何采集到的数据都存在着变化。因此，摆在生物医学工程师人员面前的任务有：

1）描述并确定噪声的特性；

2）解释包含噪声的数据；

3）确定包含噪声数据的变化规律；

4）发现规律性发生事物的普遍原理。

本章将介绍数据的归纳方法以及模型的拟合方法。

在描述统计模型时，每个用到的量称为变量。其中，自变量是可以操作或控制的变量，因变量则是测量得到的数据。以心电图记录为例，其自变量是时间，因变量则是电位。

本章主要包括如下学习内容：

1）计算测量数据的描述性统计量；

2）计算样本数据的简单统计推断；

3）应用线性最小二乘法拟合一组数据的多项式模型；

4）准确数据的多项式模型拟合；

5）基于周期函数的模型拟合。

9.2 测量、误差以及不定度

测量就是采集数据。测量仪器并不是完美无缺的，因此会引入一些误差。所谓的“真实值”是指测量仪器没有任何误差时测得的值。

但是，生物系统中不存在“真实值”。对于生物医学工程人员来说，测量的“真实值”有其他含义：生物过程固有的变异性不应该与测量误差混淆，并且要用一个量来描述这种变异性。因此，测量值实际上包含以下 3 种成分：

测量值 = 总计量 + 生物变异性 + 测量误差

检测就是要在测量误差存在的情况下找出生物过程的总计量。在测量和使用数据研究生物问题时，要牢记如下两个重要问题：

1）测量数据中是否存在系统误差和随机误差?

2）数据是否合理?

要从数据中获得有用的结论，关键是要回答好这两个问题。当然，首先，要计算多次测量数据的总和，并确定测量误差引起的数据变化特性。测量是收集数据的方法。测量误差不是系统误差就是随机误差，下面详细阐述这两种误差。

系统误差使测量数据向某个方向偏离，也就是，总是偏大或者总是偏小。例如，假设有一个电子浴室秤，其初始读数为 20 磅，那么，用这个秤测得的体重总是大于“真实”体重。增加的 20 磅就是系统误差，这种误差可以通过校正来消除。系统测量误差也称为测量偏差，偏差的来源还包括研究方案和数据采集方法中的缺陷等各种各样因素引起的单侧性误差。

随机误差是由于有限的仪器测量精度引起的测量数据的波动。这种误差是双向的，与测量偏差的单向性不同。如果你在几分钟时间内，用 10 个不同的电子浴室秤称体重，或者用同一个浴室秤称 10 次体重，在这么短的时间内你的体重是基本不变的，但是秤的读数会有所变化。这种误差无法消除，但是，通过多次测量，取平均值，可以得到比较准确的估计值。

精度和准确度是描述测量误差的两个量，读者应该弄清楚两者的区别。

精度是指多次反复测量所得到的测量值的一致程度。比如，你用浴室秤称 10 次体重，得到的读数之间的最大差别可能有 5 磅。但是，如果你用医院的人体秤进行同样的称量，得到的读数之间的最大差别可能就只有 0.5 磅了。因此，医院的人体秤精度比较高。精度由随机误差限定，可以用一组测量数据的标准差来表示其大小。

准确度是指测量值与“真实值”的符合程度。如果测量值 A 比测量值 B 更

接近“真实值”，那么，A 就比 B 准确。例如，把浴室秤调零，就是提高测量的准确度。准确度由系统误差限定。注意，如果医院里的人体秤没有经过校正，那么，这个秤就可能是高精度、低准确度的。相反，如果廉价的浴室秤测得的平均体重恰巧与真实体重很接近，那么，这个浴室秤就具有低精度、高准确度。

图 9.1 表示了准确度与精度的各种组合情况。

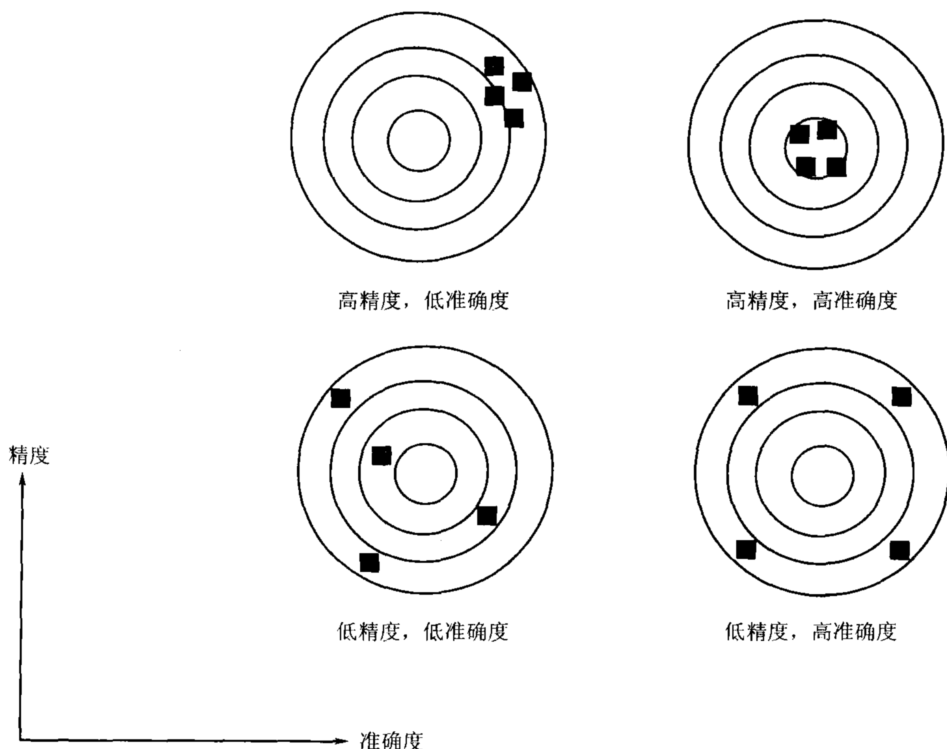


图 9.1　测量的准确度和精度

测量的重复性指标用于衡量不同次测量所得数据的一致程度，也就是，如果不同的用户用不同的测量仪器（或者在不同时间用同一台测量仪器）进行同样的测量，他们得到的测量数值应该落在重复性指标所指定的某个范围内。

分辨率是仪器可以区分的两个测量值之间的最小差别。

9.3　描述性统计学

统计就是通过对一组测量数据进行代数运算，得到一个数值。统计学用于描述数据，并进行参数估计以及统计推断（见 9.4 节）。

直方图法和数值法是用于总结归纳测量数据的两种方法。直方图是数据分布

情况的图形表示，其自变量是测量数据的可能取值范围，因变量是每个可能的测量值的发生频率。

数值法是另一种归纳测量数据分布情况的方法，它用几个描述性的统计量来表示数据分布的一些特征，包括数据所处的位置、分布的离散性、对称性以及峰的陡峭程度等。描述性统计量有以下这些：

1）描述数据分布位置的集中趋势参数，也就是，在可能的随机变量取值范围内数据的分布位置。最常用的描述集中趋势的统计量就是平均值。一组测量数据的平均值计算公式为

$$\bar{x} = \frac{\sum_{n=1}^{N} x_i}{N} \tag{9.1}$$

还有两个描述集中趋势的常用指标是中位数和众数。中位数是一组测量数据中居于中间位置的值，众数则是一组数据中出现次数最多的值。

2）描述数据差异程度的平均值的变异。描述变异特性最常用的统计量就是方差，它表示数据偏离集中趋势量的程度。一组测量数据的方差计算公式为

$$s^2 = \frac{\sum_{n=1}^{N} (x_i - \bar{x})^2}{N - 1} \tag{9.2}$$

可见，方差是测量值与其平均值之差的平均。标准差 s_x 就是方差的平方根。变异系数 cv 则是用平均值归一化处理之后的标准差，其计算公式为

$$\mathrm{cv} = \frac{s_x}{\bar{x}} \times 100\% \tag{9.3}$$

由此可见，如果分散程度相当，小平均值数据的变异系数比大平均值数据的变异系数要大。

3）偏度用于衡量数据相对于平均值的不对称变化程度，它由三阶中心矩 m_3 除以标准差的三次方得到，即

三阶中心矩

$$m_3 = \frac{\sum_{n=1}^{N} (x_i - \bar{x})^3}{(N - 1)} \tag{9.4}$$

$$偏度 = \frac{m_3}{s^3} \tag{9.5}$$

4）峰度是数据分布高耸程度的衡量指标，是四阶中心矩 m_4 与标准差四次方的比率，即

四阶中心矩

$$m_4 = \frac{\sum_{n=1}^{N} (x_i - \bar{x})^4}{(N - 1)} \tag{9.6}$$

$$峰度 = \frac{m_4}{s^4} \tag{9.7}$$

同理，可以计算更高阶的统计量。它们都是中心矩，也就是相对于平均值的矩：

$$m_k = \frac{\sum_{n=1}^{N}(x_i - \bar{x})^k}{(N-1)} \tag{9.8}$$

再用标准差的同样幂次进行归一化处理。例如，五阶中心矩的统计量就是

$$\frac{1}{s^5}\frac{\sum_{n=1}^{N}(x_i - \bar{x})^5}{(N-1)} \tag{9.9}$$

实际上，上述统计量中只有几个是常用的。表9.1列出了与这些统计量相对应的MATLAB指令。

表9.1 部分统计量的MATLAB计算指令

统 计 量	公 式	MATLAB 指令
Mean（平均值）	9.1	`mean(x)`
Median（中位数）		`median(x)`
Variance（方差）	9.2	`var(x)`
Standard Deviation（标准差）		`std(x)`
Skewness（偏度）	9.5	`moment(x,3)/std(x).^3`
Kurtosis（峰度）	9.7	`moment(x,4)/std(x).^4`

例9.1 计算MRI图像和CT图像亮度的统计量。

问题陈述：

由美国国家医学图书馆提出的可视化人体计划（Visible Human Project, VHP），旨在建立正常男性和女性详细完整的三维人体图像，该计划的长期目标是利用计算机断层成像（Computed Tomography, CT）和磁共振成像（Magnetic Resonance Images, MRI）等技术得到的详细解剖数据，将解剖知识进行符号化和文本化，如同书本等其他资源提供的解剖知识一样。VHP数据库中每个男性和女性人体都既有CT图像也有MRI图像。CT图像可以显示出骨骼、牙齿等硬组织与软组织之间的区别；而MRI图像则可以显示软组织之间的区别。

请编写一个MATLAB程序，显示VHP数据库中的CT图像和MRI图像，并统计每张图像的亮度分布数据。VHP数据库的网址为（http://www.nlm.nih.gov/research/visible/visible_human.html）。同时，利用CT图像的

描述统计量识别窦腔等解剖区域。

解：

本例题中，从 VHP 数据库网址下载了名为“head _ mri _ small”和“head _ fresh _ ct”两张图像。

MATLAB 有几条特殊的指令可用于图像的显示。首先，imread 指令是读取图像文件，并将图像数据作为数组存入 MATLAB 工作空间。image 指令用于将这个数组显示成图形。colormap 指令用于说明存储在数组中的图像亮度数据与所显示图形亮度之间的关系。

VHP 数据库中的 CT 图像和 MRI 图像都是 JPEG 格式的，为（256×256）大小的数组，亮度范围为 0~255。以下 MATLAB 指令可以显示出具有 256 个灰度级的图像：

```
v=0:(1/255):1;
map=[v;v;v]';
colormap(map);
```

如下 MATLAB 指令从 VHP 男性人体数据库读取一张 MRI 图像，存于工作空间，并显示出该图像（见图 9.2）：

```
I=imread('head_mri_small','JPEG');
image(I)
```

表 9.1 列出了计算描述统计量的一些 MATLAB 函数。但是，存储的图像是二维矩阵，二维矩阵的计算指令与一维矢量是不同的。因此，为了计算描述统计量，必须先用如下指令将图像矩阵的数据转化成一维矢量：

```
[rI,cI]=size(I);
II=reshape(I,rI*cI,1);
meanI=mean(II);
varI=var(II);
stdI=std(II);
fprintf('\nAverage intensity\t% f\nVariance\t% f\nStandard
Deviation\t% f\n',meanI,varI,stdI);
```

VHP 图像为 256 行乘 256 列，这里把图像数据转化成了（65536×1）的一维矢量，用于计算图像亮度的平均值、方差和标准差，并用 fprintf 格式化指令输出这 3 个统计量的值。

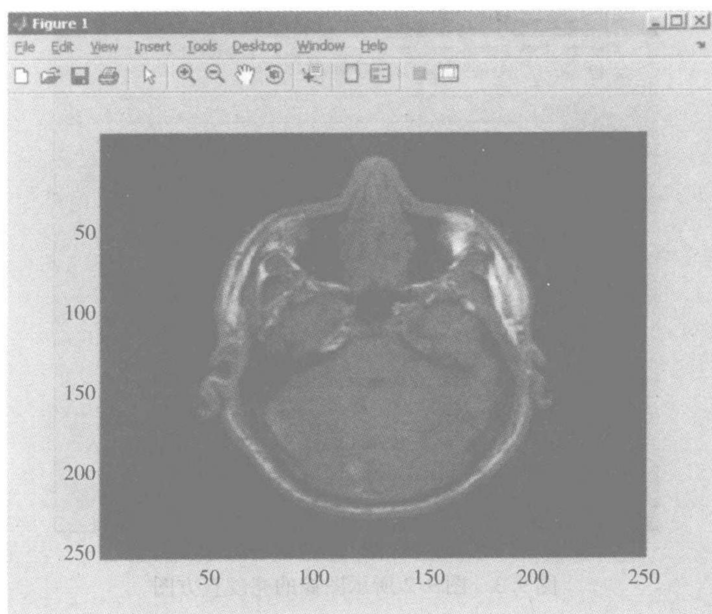


图 9.2　用 MATLAB 指令显示的 MRI 图像

用直方图（见图 9.3）可以将图像的亮度分布情况用图形表示出来，下面的 MATLAB 指令就可以方便地计算和显示直方图：

```
h=histc(II,0:255);
figure(2)
plot(0:255,h,'.',[meanI;meanI;meanI],[100;500;1000],'s')
```

其中的 plot 指令把平均值所处的位置用 3 个重叠的小正方形表示出来。

用同样的方法可以显示和处理 CT 图像，输出结果如图 9.4 和 9.5 所示：

```
J=imread('head_fresh_ct','JPEG');
figure(3)
colormap(map)
image(J)
[rJ,cJ]=size(J);
JJ=reshape(J,rJ*cJ,1);
meanJ=mean(JJ);
varJ=var(JJ);
stdJ=std(JJ);
```

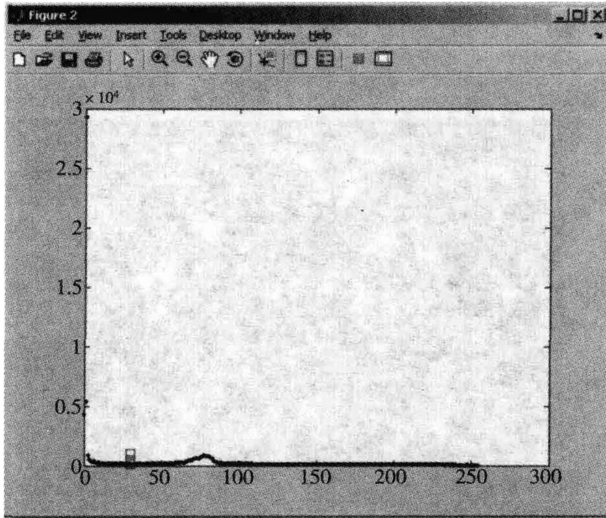


图 9.3 图 9.2 所示图像的亮度直方图

```
h = histc(JJ,0:255);
figure(4)
plot(0:255,h,'.',[meanJ;meanJ;meanJ],[100;500;1000],'s')
```

有时，可以用描述统计量识别和定位图像中一些重要区域。例如，CT 图像中亮度等于或接近于 0 的暗区表示的是窦腔。用图像的平均亮度值作为阈值，可以分割出图像中可能对应于窦腔的区域。程序如下：

```
figure(5)
colormap(map)
TJ = J;
TJ(J < meanJ) = 0;
TJ(TJ > 0) = 255;
image(TJ)
```

其中，利用 MATLAB 的逻辑索引建立了一个新矩阵 TJ，表示 CT 图像 J 经过阈值处理之后的图像。

图 9.6 中显示的三大块黑色区域就对应于 3 个窦腔，靠近图像上方的两个为上颌窦，上颌窦下方的是蝶窦。比较图 9.4 和图 9.6 两张图，可见，利用图像亮度平均值进行阈值处理的方法进行图像窦腔的识别，其结果并不是非常理想。这类问题属于图像处理以及医学图像分析领域的研究课题。

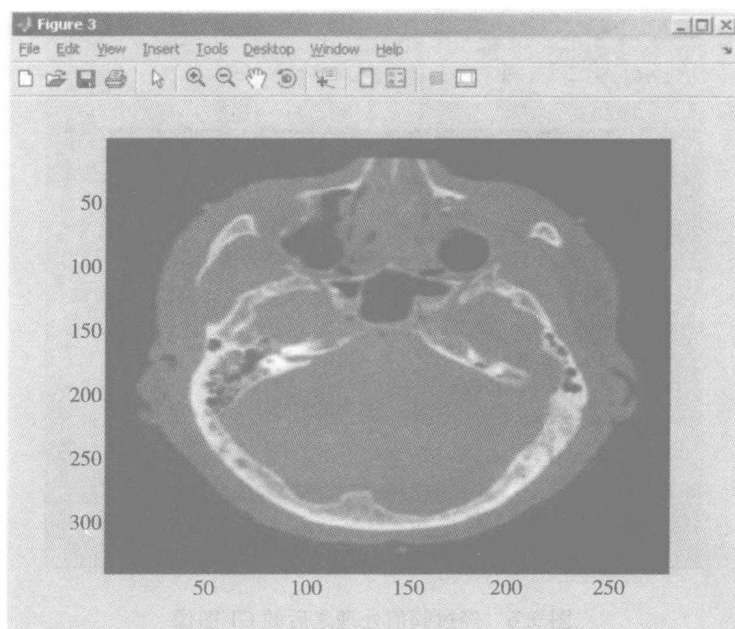


图 9.4　用 MATLAB 指令显示的 CT 图像

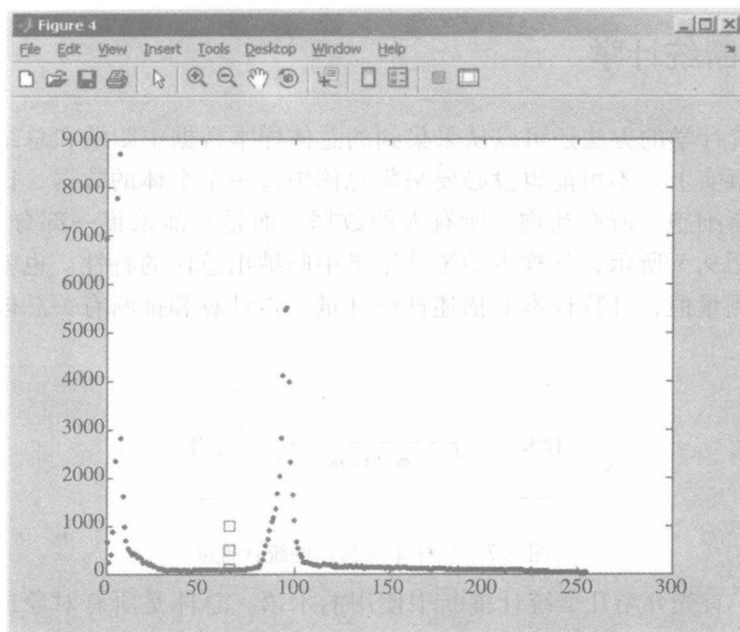


图 9.5　图 9.4 所示图像的亮度直方图

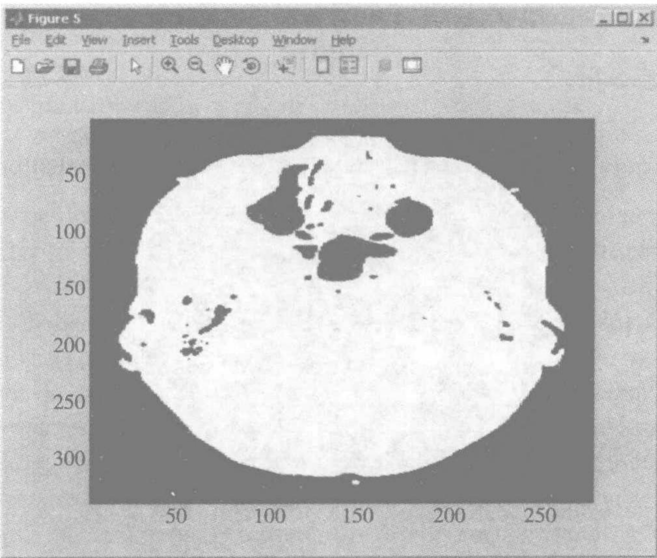


图 9.6 经过阈值处理之后的 CT 图像

如果想了解有关可视化人体计划 VHP 的详细内容，请访问 NLM 网站（http：//www. nlm. nih. gov/）。

9.4 推断统计学

利用统计学的方法还可以从采集到的总体样本数据中归纳和总结出该总体的特性。事实上，不可能也没必要采集总体中每一个个体的数据，比如，不可能得到所有细胞、所有动物、所有人的数据。而是只能采集一部分样本数据，然后，如图 9.7 所示，从样本的统计结果中归纳出总体的特性。也就是，先收集样本的测量值，计算样本的描述性统计量，再计算和推断有关总体的一般特性。

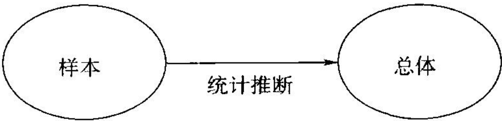


图 9.7 从样本到总体的统计推断

这里，首先介绍几个统计推断中使用的术语。总体是所有对象或物体的集合，如人、动物、细胞或分子。样本是总体的子集，由收集到的测量值构成。总体参数描述总体的特性，测量时这些参数是未知的。样本统计量则是根据所采集

的样本数据计算得到的总体参数的估计值。如果用不同的样本进行计算，样本统计量彼此之间会有些变化，样本分布就是描述不同样本集合的样本统计量的变化规律。

统计推断用于确定某个样本统计量是否为总体参数的“好”估计值。每个估计量都有两个特性：“值”和“信度”。“值”说明了样本统计量预测总体参数的好坏。“信度”则表示样本统计量用于所有可能的同类测量时的可信程度。如果测量中采用了制定完善的标准过程，那么信度表示样本测量中所获得的结果在其他所有可能的样本测量中同样成立的可能性。通过统计推断的分析，我们才能得出结论，说明测量数据的变化是由于生物变异性或测量误差引起的，而不是由某个未知因素引起。

变量之间关系的统计推断方法有两种，即参数估计和假设检验。参数估计就是根据样本数据计算出某个参数值（即上述的“值”），并且给出该参数值可能的变化范围，即置信区间（上述的“信度”）。如果置信区间较大，则该区间包含总体参数值的可能性就大。如果置信区间较小，那么总体参数的估计精度就较高。

从样本统计量计算总体参数置信区间的一般公式为

$$\text{统计量} \pm z \cdot s_{\text{统计量}} \tag{9.10}$$

式中 z——根据样本数据以及所要求的置信水平得到的值，z 值可以从书中或网上查表获得。

例如，总体均值的置信区间为

$$M - z \cdot s_M \leqslant \mu \leqslant M + z \cdot s_M \tag{9.11}$$

式中 M——样本均值。

如果除了均值之外，标准差也是通过计算获得的，那么上式中的 z 就要用 t 代替，即

$$M - t \cdot s_M \leqslant \mu \leqslant M + t \cdot s_M \tag{9.12}$$

例 9.2 用样本数据估计总体的均值。

正态分布（即高斯分布）可以用均值 μ 和方差 σ 这两个总体参数描述。假设有一个正态分布随机变量的样本，标准差已知，请计算均值的置信区间。要求均值包含在该置信区间中的概率有 95%。

解：

如果正态分布的 $\mu=0$ 且 $\sigma=1$，则称为标准正态分布。MATLAB 的如下指令产生标准正态分布的 100 个样本：

```
d=random('normal',0,1,[100  1])
```

当 $\sigma=1$ 时，如果置信度为 95%，则 $z=1.96$。置信区间计算如下：

```
M=mean(d);
sM=std(d);
low=M-1.96;
high=M+1.96;
fprintf('The confidence interval is (% f,% f)\n',low,high)
```

用不同的样本重复执行这段程序，可以得到不同的结果。某输出结果为

```
The confidence interval is (-2.121312,1.798688)
```

当样本量越来越大时，置信区间会趋近于（-1.96，1.96）。

另外，如果 σ 未知，要从样本中估计其值，则可以用如下 MATLAB 程序计算置信区间：

```
M=mean(d);
s=std(d);
sM=s/sqrt(100);
low=M-1.9842*sM;
high=M+1.9842*sM;
fprintf('The confidence interval is(% f,% f)\n',low,high)
```

其中 $t=1.9842$，由查表获得。这段程序的输出结果是

```
The confidence interval is (-0.235331,0.151654)
```

统计推断的另一种方法是假设检验，用于比较关于总体的某两个假设的真实程度。通过测量的样本数据考察自变量的作用效果时，很重要的一点是要有足够的置信度来说明这种效果确实是由自变量的变化引起的，而不是由测量误差等其他变化引起的。假设检验用于拒绝两个假设之一的零假设而接受另一个备择假设。

备择假设是有关总体参数的一个假设，它假定自变量对于因变量确实具有作用效果。而零假设通常是与备择假设相对立的假设，设置零假设是为了考察样本数据与该假设是否存在矛盾。假设检验一般分为 4 步：

1）确定零假设和备择假设。

2）选择所需要的置信水平，也称为显著性水平。置信水平说明了拒绝零假设并接受备择假设的可信度。

3）计算样本统计量用于估计总体参数。

4）确定在零假设成立的情况下该样本统计量可能出现的概率（即 p 值），并将此 p 值与显著性水平进行比较，如果 p 值小于显著性水平，则拒绝零假设，接受备择假设。

这里只是非常简单地介绍了统计推断和假设检验。下面说明生物医学工程中常用的两种检验方法，即学生 t 检验和相关。

假设检验中用到的置信区间公式与上述一般公式基本相同。一旦选定了零假设，总体参数就确定了。设定显著性水平并完成实验之后，就可以计算出样本统计量 M（如果需要还要计算标准差 s_M）。如果样本方差已知，就根据 z 表计算 p 值；如果样本方差未知，则根据 t 表计算 p 值。根据计算结果，如果置信区间不等式不成立，那么就拒绝零假设。

学生 t 检验简称 t 检验，是用于描述两个总体均值之间关系的假设检验。它的假设是要么两个总体均值相同（即零假设），要么两个总体均值不相同（即备择假设）。实际应用中，样本统计量经常设为两个样本均值之差，于是，零假设就是均值之差等于 0，而备择假设就是两个均值之差不等于 0。例如，如果样本统计量为 $M_1 - M_2$，t 检验的两个假设为

$$\begin{aligned} H_0 &: \mu_1 = \mu_2 \\ H_1 &: \mu_1 \neq \mu_2 \end{aligned} \tag{9.13}$$

重排置信区间公式，可求得 t 值：

$$t = \frac{(M_1 - M_2) - (\mu_1 - \mu_2)}{s_{M_1 - M_2}} \tag{9.14}$$

将零假设的 $\mu_1 = \mu_2$ 代入，可得

$$t = \frac{(M_1 - M_2)}{s_{M_1 - M_2}} \tag{9.15}$$

其中 $s_{M_1 - M_2}$ 可由下式求得

$$s_{M_1 - M_2} = \sqrt{\frac{s_1^2}{N_1} + \frac{s_2^2}{N_2}} \tag{9.16}$$

另一种常用的检验方法是相关，它是自变量和因变量之间线性关系的假设检验。当相关系数 r 等于 -1 时，表示负线性关系；等于 0 时，表示没有线性关系；等于 1 时，表示正线性关系。因此，相关系数 r 在 -1—0—1 之间变化。通常用 r^2 而不用 r。

假设 X 和 Y 分别为自变量和因变量。如果样本方差已知，则相关系数 r 为

$$r = \frac{\sum_{i=1}^{N} X_i Y_i - \sum_{i=1}^{N} X_i \sum_{i=1}^{N} Y_i}{\sigma_X \sigma_Y} \tag{9.17}$$

如果样本方差未知，则相关系数 r 为

$$r=\frac{\sum_{i=1}^{N}X_iY_i-\frac{\sum_{i=1}^{N}X_i\sum_{i=1}^{N}Y_i}{N}}{\sqrt{\left(\sum_{i=1}^{N}X_i^2-\frac{\left(\sum_{i=1}^{N}X_i\right)^2}{N}\right)\left(\sum_{i=1}^{N}X_i^2-\frac{\left(\sum_{i=1}^{N}X_i\right)^2}{N}\right)}} \tag{9.18}$$

注意：上式中随机变量 X 和 Y 是可以互换的。这表示用相关系数测量的相关性是一种对称关系，这意味着相关性强并不一定存在因果关系，也就是一个变量的变化并不一定是对于另一个变量变化的响应。

虽然上式是最常用的相关公式，但是它并不能反映变量之间的非线性关系。还有其他类型的相关方法可以处理其他类型的变量、非线性关系、多重相关性、以及两个或更多自变量与一个因变量之间的关系等。

例 9.3 DNA 芯片数据分析中的假设检验

DNA 芯片用于研究不同实验条件下基因的相对表达水平，其中常用的一个系统是酿酒酵母基因芯片。

DNA 芯片的数据可以由作者提供，也可以从 NIH 基因表达网站（http：//www. ncbi. nlm. nih. gov/genome/guide/human/resources. shtml）获得。MATLAB 的生物信息数据库中有一个样本库。请从这个样本库中挑选两个基因，并检验两个基因的表达模式是否相关。

解：

如果有 MATLAB 生物信息工具箱，就可以直接调入 yeastdata. mat 变量；如果没有这个工具箱，可以用序列号 GSE28 从 NIH 网站获得完整的数据，网址为：http：//www. ncbi. nlm. nih. gov/entrez/query. fcgi？db = gds&cmd = search&term = GSE28

执行如下 MATLAB 指令之后：

```
load yeastdata.mat
```

工作空间的 yeastdata 数组中存放的就是芯片的基因表达水平，元胞数组 genes 中存放的是这个数据库的 6400 个基因的有关数据。yeastdata 数组的每一行存放了 7 个时间点上基因的相对表达水平，实际的时间值则存放在工作空间的“times”变量中。以下程序选择了第 20 和第 21 两个基因：

```
figure(1)
plot(times,yeastvalues(20:21,:))
```

如图 9.8 所示是这两个基因的相对表达水平的输出结果，两者似乎是相关的。设定“相关”为备择假设，“无线性相关”为零假设。利用 MATLAB 的 corr 函数可以计算两个基因表达曲线的相关系数，指令如下，两组表达数据作为 corr 函数的参数：

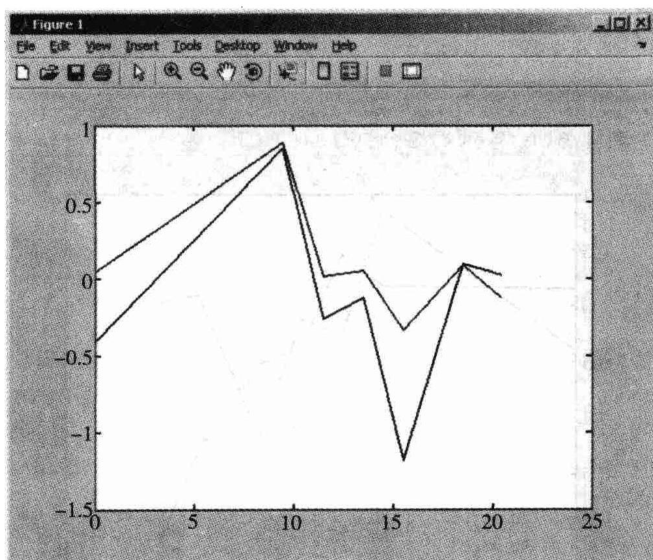


图 9.8[⊖]　第 20 和第 21 基因的相对表达水平

```
corr(yeastvalues(20,:)',yeastvalues(21,:)')
```

得到的相关系数 r 为 0.8693。零假设的相关系数为 0，用如下公式计算 t 值：

$$t = \frac{r}{\sqrt{(1 - r^2)/(N - 2)}}$$

程序为

```
N = length(times);
t = r/sqrt((1 - r^2)/(N - 2))
```

对于第 20 和第 21 基因，t 值为 3.9331。现在，取显著性水平为 0.05；由于有 7 个时间点上的数据，因此，自由度的值为 5。从 t 值表中查到这些条件所对应的临界值 t 为 2.5706。3.9331 大于 2.5706，因此，拒绝零假设，接受备择假设，

⊖ 原书此图号为 9.7，与前图号重复，此图号现改为 9.8，后各图顺改。——译者注

即两个基因的表达模式相关。

下面选择第 20 和第 30 两个基因。图 9.9 显示了两者的表达模式，程序如下：

```
r=corr(yeastvalues(20,:)',yeastvalues(30,:)')
t=r/sqrt((1-r^2)/(N-2))
```

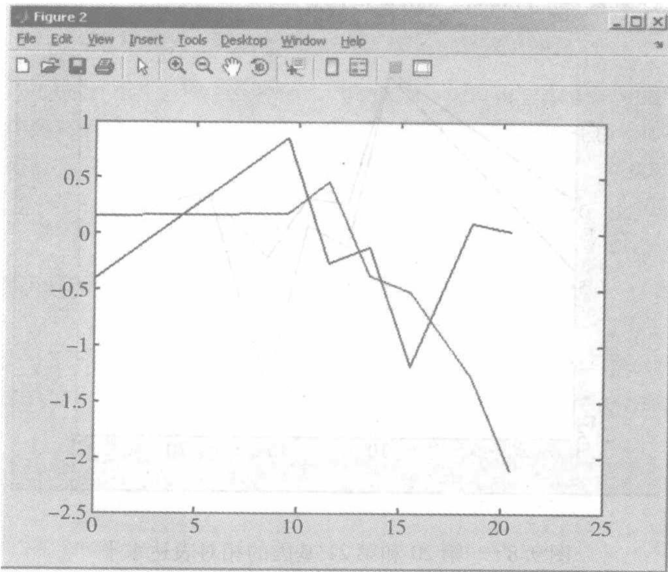


图 9.9 第 20 和第 30 基因的相对表达水平

求得的相关系数为 -0.0150，其中的负号表示负相关。t 值为 -0.0335，小于查表得到的临界值 2.5706，因此，不能拒绝“无相关”这个零假设。

这里所谓“不能拒绝零假设”并不是指可以接受零假设，因为“拒绝零假设”的含义是：如果这种样本之间的相关性是由于碰巧才发生的，那么这种碰巧发生的概率小于 0.05。

9.5 最小二乘法建模

最小二乘法是用解析数学模型拟合含噪数据的一种数值方法，就是用测得的数据计算模型的参数。

假设因变量的测量值是某个自变量的函数，比如，随时间变化的电位，则拟合测量数据的模型可以设为：

$$Y = f(X) + \varepsilon \tag{9.19}$$

式中　X——自变量；

Y——因变量；

ε——噪声。

假设有一组测量数据 (x_i, y_i)，并且，自变量 X 与因变量 Y 之间具有线性关系，那么，如果没有噪声干扰，用如下模型

$$Y = aX + b \tag{9.20}$$

可以完全拟合数据。

最小二乘法计算模型参数值的原理，是使模型数据与测量数据之间总的差值最小化。当测量中存在误差时，对于以上线性模型，测量值 y_i 与模型计算值 $(ax_i + b_i)$ 之间就会有差值：

$$d_i = y_i - (ax_i + b_i) \tag{9.21}$$

求模型拟合的总误差时，最好不要用带正负号的差值，而要用差值的平方，这样，其总和才能更好地反映拟合的误差，即

$$E_T = \sum_{i=1}^{n} (y_i - (ax_i + b))^2 \tag{9.22}$$

这里的数值问题就是要确定参数 a 和 b 的值，使得 E_T 最小化。使 E_T 相对于每个参数的偏导数等于 0，即

$$\begin{aligned} \frac{\partial E_T}{\partial a} &= -2\sum_{i=1}^{n} (y_i - (ax_i + b))x_i = 0 \\ \frac{\partial E_T}{\partial a} &= -2\sum_{i=1}^{n} (x_i y_i - ax_i^2 - bx_i) = 0 \end{aligned} \tag{9.23}$$

且

$$\frac{\partial E_T}{\partial b} = -2\sum_{i=1}^{n} (y_i - (ax_i + b)) = 0 \tag{9.24}$$

就可以使 E_T 最小。

由此可以得到求解参数 a 和 b 的方程组：

$$\begin{aligned} \frac{\partial E_T}{\partial a} &= -2\sum_{i=1}^{n} (x_i y_i - ax_i^2 - bx_i) = 0 \\ \frac{\partial E_T}{\partial b} &= -2\sum_{i=1}^{n} (y_i - ax_i - b) = 0 \end{aligned} \tag{9.25}$$

除去系数 -2，并重排，可得

$$\begin{aligned} a\sum_{i=1}^{n} x_i^2 + b\sum_{i=1}^{n} x_i &= \sum_{i=1}^{n} x_i y_i \\ a\sum_{i=1}^{n} x_i + b\sum_{i=1}^{n} 1 &= \sum_{i=1}^{n} y_i \end{aligned} \tag{9.26}$$

可以进一步化为

$$
\begin{aligned}
b\sum_{i=1}^{n}1 + a\sum_{i=1}^{n}x_i &= \sum_{i=1}^{n}y_i \\
b\sum_{i=1}^{n}x_i + a\sum_{i=1}^{n}x_i^2 &= \sum_{i=1}^{n}x_iy_i
\end{aligned}
\tag{9.27}
$$

写成矩阵矢量形式，则为

$$
\begin{bmatrix} n & \sum_{i=1}^{n}x_i \\ \sum_{i=1}^{n}x_i & \sum_{i=1}^{n}x_i^2 \end{bmatrix}\begin{bmatrix} b \\ a \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n}y_i \\ \sum_{i=1}^{n}x_iy_i \end{bmatrix}
\tag{9.28}
$$

这些方程组称为正规方程组，是建立最小二乘问题的标准形式。是线性代数方程组，可以用本书第 4 章介绍的方法求解。

例 9.4 一阶多项式（即直线）的最小二乘法拟合

利用最小二乘法实现一组测量数据 (x_i, y_i) 与直线的拟合，数据为

(1,4),(3,5),(5,6),(7,5),(10,8),(12,7),(13,6),(16,9),(18,12),(20,11)

并用 MATLAB 程序求解线性方程的系数。

解:

用如下两个数组分别存放自变量和因变量的测量值:

```
x=[1  3  5  7  10  12  13  16  18  20];
y=[4  5  6  5  8  7  6  9  12  11];
```

用如下 MATLAB 矩阵表示本题的正规方程组，并计算矩阵的值:

```
A=[length(x),sum(x);sum(x),sum(x.^2)]
A=
          10        105
          105       1477
```

式（9.28）的右边为

```
c=[sum(y);sum(x.*y)]
c=
     73
    906
```

这样，就可以求得系数:

```
z=A^-1*c
z=
      3.3888
      0.3725
```

也就是，拟合直线的截距为3.3888，斜率为0.3725。

以下MATLAB指令可以方便地作图，同时显示测量数据以及拟合直线：

```
plot(x,y,'o',x,z(1)+z(2)*x)
```

其中，测量数据用圆圈表示，直线就是拟合模型。作图指令中的前两个参数分别为测量数据的自变量和因变量，第三个参数'o'表示数据点显示形式的符号，它表示用圆圈显示数据点。最后两个参数为模型拟合线的自变量和因变量，这里的因变量不是测量数据，而是模型的数据。计算式 z(1) + z(2) * x 是一个矢量，其值的计算是 x 的每个元素乘以估计的斜率值，再加上估计的截距值。图9.10所示为该MATLAB作图指令的输出结果。

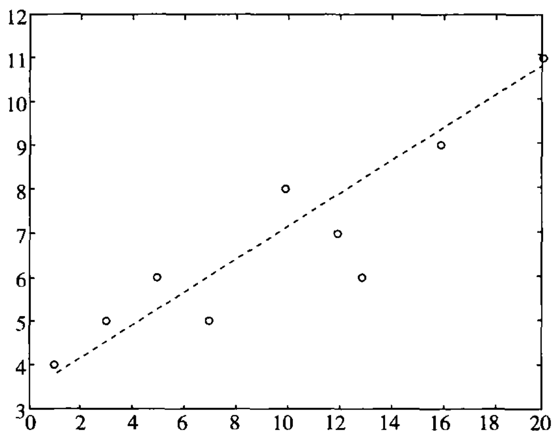


图9.10　测量数据及其拟合模型

例9.5　三阶多项式的最小二乘法拟合

请建立用三阶多项式拟合 (x_i, y_i) 数据组的最小二乘法。

解：

三阶多项式的形式为

$$y_i = a_0 + a_1 x_i + a_2 x_i^2 + a_3 x_i^3$$

同理，每个测量数据与拟合数据差值的平方为

$$(y_i - (a_0 + a_1 x_i + a_2 x_i^2 + a_3 x_i^3))^2$$

对于每个参数 a_i 求偏导数之后，得到的正规方程组为

$$\begin{bmatrix} n & \sum x_i & \sum x_i^2 & \sum x_i^3 \\ \sum x_i & \sum x_i^2 & \sum x_i^3 & \sum x_i^4 \\ \sum x_i^2 & \sum x_i^3 & \sum x_i^4 & \sum x_i^5 \\ \sum x_i^3 & \sum x_i^4 & \sum x_i^5 & \sum x_i^6 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} \sum y_i \\ \sum x_i y_i \\ \sum x_i^2 y \\ \sum x_i^3 y \end{bmatrix}$$

参数 a_i 的解即为

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} n & \sum x_i & \sum x_i^2 & \sum x_i^3 \\ \sum x_i & \sum x_i^2 & \sum x_i^3 & \sum x_i^4 \\ \sum x_i^2 & \sum x_i^3 & \sum x_i^4 & \sum x_i^5 \\ \sum x_i^3 & \sum x_i^4 & \sum x_i^5 & \sum x_i^6 \end{bmatrix}^{-1} \begin{bmatrix} \sum y_i \\ \sum x_i y_i \\ \sum x_i^2 y \\ \sum x_i^3 y \end{bmatrix}$$

将非线性方程转化为线性方程，应用同样的方法还可以进行非线性模型的拟合。

例 9.6 非线性模型的最小二乘法拟合

请建立用 $y = ae^{bx}$ 模型拟合（x_i，y_i）数据组的最小二乘法。

解：

首先，对方程 $y = ae^{bx}$ 两边求自然对数，得到：

$$\ln y = \ln a + bx$$

设 $a_0 = \ln a$，$a_1 = b$，则问题就转化为求解线性最小二乘法。其正规方程组为

$$\begin{bmatrix} n & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^2 \end{bmatrix} \begin{bmatrix} \ln a \\ b \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} \ln y_i \\ \sum_{i=1}^{n} x_i \ln y_i \end{bmatrix}$$

这样，就可以先求得 ln a 和 b 的值，然后，很容易得到非线性形式的系数 a 的值。

用同样的方法，可以拟合多变量的模型，如下例所述。

例 9.7 多变量模型的最小二乘法拟合

请建立一个多变量模型拟合数据组（$x_{1,i}$，$x_{2,i}$，y_i），其中，有 x_1 和 x_2 两个自变量，以及 y 一个因变量。

解：

设模型的形式为

$$y = a_0 + a_1 x_1 + a_2 x_2 + \varepsilon$$

式中　ε——测量噪声。

其正规方程组为

$$\begin{bmatrix} n & \sum x_{1i} & \sum x_{2i} \\ \sum x_{1i} & \sum x_{1i}^2 & \sum x_{1i}x_{2i} \\ \sum x_{2i} & \sum x_{1i}x_{2i} & \sum x_{2i}^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \sum y_i \\ \sum x_{1i}y_i \\ \sum x_{2i}y_i \end{bmatrix}$$

9.6　曲线拟合

最小二乘法采用的模型可以有很多种不同的形式。上节中列举的如下拟合模型

$$\begin{aligned} y &= a_0 + a_1 x \\ y &= a_0 + a_1 x + a_2 x^2 + a_3 x^3 \\ y &= a\mathrm{e}^{bx} \\ y &= a_0 + a_1 x_1 + a_2 x_2 \end{aligned} \tag{9.29}$$

只是可能选取的大量解析模型中的少数几个例子。

模型还用于预测对应于某个输入值的输出值，特别是预测对应于某个自变量的没有采集到的输出数据值。模型的预测有内插和外插两种。

假设我们有一组数据，范围是 (x_1,y_1) ~ (x_n,y_n) ，要预测的是某给定值 x 上的输出 y，那么，当 $x_1<x<x_n$ 并且 x 不等于 x_i 时，这种预测就是内插；而当 $x<x_1$ 或 $x>x_n$ 时，就是外插。

如果数据是在存在噪声的情况下采样得到的，那么，就要用 9.4 节介绍的最小二乘法计算模型的参数（即系数）。另一方面，如果已知精确的数据值，也就是数据是由精确查表得到的，那么，就不需要用最小二乘法了，可以用第 6 章介绍的格雷戈里-牛顿（Gregory-Newton）插值公式和拉格朗日（Lagrange）插值多项式来拟合精确的数据，并进行内插或外插计算。

9.6.1　拉格朗日插值多项式

原则上，经过 $n+1$ 个数据点的 n 阶多项式是惟一的，例如，用两点数据可以估计如下一阶多项式

$$y = a_0 + a_1 x \tag{9.30}$$

的两个系数 a_0 和 a_1。

假设两点数据的值分别为 (x_0,y_0) 和 (x_1,y_1) ，这个一阶多项式必须正好经过这两点，也就是，对于求得的插值多项式 $P(x)$ ，必须有 $P(x_0)=y_0$ ，$P(x_1)=y_1$ 。显然，多项式

$$P_1(x) = \frac{(x-x_1)}{(x_0-x_1)}y_0 + \frac{(x-x_0)}{(x_1-x_0)}y_1 \tag{9.31}$$

满足这些条件。当 $x=x_0$ 时，第二项为 0，就有 $P_1(x_0)=y_0$ ；而 $x=x_1$ 时，第一

项为0，就有 $P_1(x_1)=y_1$。经过3个数据点 (x_0,y_0)、(x_1,y_1) 和 (x_2,y_2) 的插值多项式 $P_2(x)$ 为：

$$P_2(x)=\frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}y_0+\frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)}y_1+\frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}y_2 \tag{9.32}$$

$P_2(x)$ 有3个线性组合项，每项都是二阶多项式，且在两个数据点上值为0，而在其余一个数据点上系数值为1。将这3项记为

$$\begin{aligned}L_{2,0}(x)&=\frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}\\L_{2,1}(x)&=\frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)}\\L_{2,2}(x)&=\frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}\end{aligned} \tag{9.33}$$

则插值多项式 $P_2(x)$ 成为

$$P_2(x)=\sum_{k=0}^{2}y_kL_{2,k}(x) \tag{9.34}$$

这就是拉格朗日插值多项式，其一般形式为

$$P_n(x)=\sum_{k=0}^{n}y_kL_{n,k}(x) \tag{9.35}$$

对于 n 阶插值多项式，有

$$L_{n,k}(x)=\frac{(x-x_0)(x-x_1)\cdots(x-x_{k-1})(x-x_{k+1})\cdots(x_k-x_n)}{(x_k-x_0)(x_k-x_1)\cdots(x_k-x_{k-1})(x_k-x_{k+1})\cdots(x_k-x_n)} \tag{9.36}$$

9.6.2 牛顿差商插值多项式

拉格朗日插值法并不是确定插值多项式的惟一方法，由数据表求取插值多项式的方法称为差商法。差商法也用于求函数导数和积分的近似值，并用于求微分方程的近似解。

差商用如下特定的符号表示，设 $f[x_i]=f(x_i)$，则

$$f[x_i,x_{i+1}]=\frac{f(x_{i+1})-f(x_i)}{(x_{i+1}-x_i)} \tag{9.37}$$

就是关于 x_i 和 x_{i+1} 的一阶有限差商。同理，利用 $k-1$ 阶差商 $f[x_i,x_{i+1},x_{i+2},\cdots,x_{i+k-1}]$ 和 $f[x_{i+1},x_{i+2},\cdots,x_{i+k}]$，可以定义 $x_i,x_{i+1},x_{i+2},\cdots,x_{i+k}$ 点上的 k 阶差商为

$$f[x_i,x_{i+1},x_{i+2},\cdots,x_{i+k}]=\frac{f[x_i,x_{i+1},x_{i+2},\cdots,x_{i+k-1}]-f[x_{i+1},x_{i+2},\cdots,x_{i+k}]}{x_{i+k}-x_i} \tag{9.38}$$

这样，拉格朗日插值多项式 $P_n(x)$ 也可以写为

$$P_n(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \cdots + a_n(x - x_0)\cdots(x - x_{n-1}) \tag{9.39}$$

因为 $P_n(x_0) = f(x_0)$，可得系数 a_0 的值为 $a_0 = f(x_0) = f[x_0]$。同样：

$$P_n(x_1) = f(x_1) = a_0 + a_1(x_1 - x_0) \tag{9.40}$$

可得

$$a_1 = \frac{f(x_1) - f(x_0)}{(x_1 - x_0)} = f[x_0, x_1] \tag{9.41}$$

于是，插值多项式可以写为

$$\begin{aligned} P_n(x) = {} & f[x_0] + f[x_0, x_1](x - x_0) + a_2(x - x_0)(x - x_1) + \cdots \\ & + a_n(x - x_0)\cdots(x - x_{n-1}) \end{aligned} \tag{9.42}$$

用同样的方法可以确定其余系数的值。即

$$a_k = f[x_0, x_1, \cdots, x_k] \tag{9.43}$$

这样，插值多项式 $P_n(x)$ 就成为

$$P_n(x) = f[x_0] + \sum_{k=1}^{n} f[x_0, \cdots, x_k](x - x_0)\cdots(x - x_{k-1}) \tag{9.44}$$

这就是牛顿差商插值多项式。关于差商的计算公式请参阅第6章。

9.6.3 样条

拉格朗日插值多项式和牛顿均差插值多项式基于同样的原理，如果要建立一个 n 阶拟合函数就需要有 $n+1$ 个数据点。实际上，利用其他约束条件，可以减少所需数据的数量。样条算法求取多项式近似公式时，就是用相关的约束条件代替部分数据点。

有时，希望用较低阶的多项式拟合 $n+1$ 个数据点，因为，较高阶的多项式会反映出所有的变化。例如，对于医学图像等，希望得到光滑的表面，小的变化反而会降低图像的质量。样条是用多个低阶多项式来拟合整个数据组，其中，每个多项式只拟合数据的一部分，一组低阶多项式就可以拟合全部数据。低阶多项式比较光滑，因此，就可以产生较好的图像视觉效果。拉格朗日插值多项式和牛顿均差插值多项式拟合数据的隐含约束条件是多项式必须经过所有 $n+1$ 个数据点。假设只有两个数据点，那么，根据这个原则，插值的结果只能是这两个点之间连成的直线函数。但是，如果加上拟合曲线光滑拼接的约束条件，那么，不增加数据点，也可以用二阶、甚至三阶函数来拟合两个数据点。

假设要用二阶函数（或三阶函数）拟合 $n+1$ 个数据点中两两相邻数据点形成的 n 个数据对，每个二阶方程：

$$f_i(x) = a_{i,0} + a_{i,1}x + a_{i,2}x^2 \tag{9.45}$$

需要确定3个未知系数 $a_{i,0}$、$a_{i,1}$ 和 $a_{i,2}$。全部拟合中一共有 $3n$ 个未知系数，因此，需要 $3n$ 个方程来求解这些未知系数。除了数据点以外，还有如下这些一般规则用于确定样条多项式的系数：

1）相邻多项式在同一个内部数据点上的函数值必须相等，且都等于给定的数据值。将这条规则应用于 $n-1$ 个内部数据点，就得到 $2n-2$ 个条件，每个数据点上有两个等式，即

$$\begin{aligned} f(x_{i-1}) &= a_{i-1,0} + a_{i-1,1}x_{i-1} + a_{i-1,2}x_{i-1}^2 \\ f(x_{i-1}) &= a_{i,0} + a_{i,1}x_{i-1} + a_{i,2}x_{i-1}^2 \end{aligned} \tag{9.46}$$

2）第一个和最后一个多项式必须分别经过首尾两端的数据点。由此产生两个条件，即

$$\begin{aligned} f_1(x_0) &= a_{1,0} + a_{1,1}x_0 + a_{1,2}x_0^2 \\ f_n(x_n) &= a_{n,0} + a_{n,1}x_n + a_{n,2}x_n^2 \end{aligned} \tag{9.47}$$

3）内部数据点上的一阶导数必须连续。由此产生 $n-1$ 个条件，形式为

$$a_{i-1,1} + a_{i-1,2}x_{i-1} = a_{i,1} + a_{i,2}x_{i-1} \tag{9.48}$$

4）任意选择首尾两个数据点中的一个，使这点上的二阶导数为0。例如，设第一个多项式在第一个数据点上的二阶导数为0，则有

$$a_{1,2} = 0 \tag{9.49}$$

这样，上述约束条件的总数为

规　　则	条　件　数	规　　则	条　件　数
1	$2n-2$	3	$n-1$
2	2	4	1
总计		$3n$	

因此，就有足够的条件求解各个二阶多项式的系数。假设给定的数据为 (x_0, y_0) ~ (x_n, y_n)，利用以上这些方程求解各个样条函数系数的线性系统就是

$$\begin{bmatrix} 1 & x_0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 1 & x_1 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 1 & x_1 & x_1^2 & \cdots & 0 & 0 & 0 \\ 1 & x_1 & -1 & -x_1 & 0 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & 1 & x_n & x_n^2 \end{bmatrix} \begin{bmatrix} a_{1,0} \\ a_{1,1} \\ a_{2,0} \\ a_{2,1} \\ a_{2,2} \\ \vdots \\ a_{n,0} \\ a_{n,1} \\ a_{n,2} \end{bmatrix} = \begin{bmatrix} f(x_0) \\ f(x_1) \\ f(x_1) \\ 0 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ f(x_n) \end{bmatrix} \tag{9.50}$$

注意，因为规则 4 已经设定 $a_{1,2}=0$，因此，系统中没有这个系数。第一个方程和最后一个方程由规则 2 得到，第二个和第三个方程由规则 1 得到，第四个方程则由规则 3[⊖]得到。对于 x_1 到 x_{n-1} 的每个内部数据点重复应用 1 和 3 两条规则[⊜]，得到系统中其余的方程。利用第 4 章介绍的那些方法都可以求解这个线性系统，得到系数 $a_{i,j}$ 的值。

利用同样的方法，可以确定三阶样条函数的系数，只是需要增加一条规则，规定内部数据点上的二阶导数必须连续。因此，三阶样条函数的规则为

1）相邻多项式在内部数据点上的函数值必须相等，且都等于给定的数据值。

2）第一个和最后一个多项式必须分别经过首尾两端的数据点。

3）内部数据点上的一阶导数必须连续。

4）内部数据点上的二阶导数必须连续。

5）首尾两个数据点上的二阶导数都为 0。

这样，约束条件的总数为

规　则	条 件 数	规　则	条 件 数
1	$2n-2$	4	$n-1$
2	2	5	2
3	$n-1$	总计	$4n$

这个条件数与需要求解的所有三阶样条函数系数的个数相等。

MATLAB 有内置函数可以用于计算样条函数的系数，并给出样条函数。

例 9.8　MALDI-TOF 质谱数据的重采样和基线校正。

质谱是用于分析和表征大分子化合物的一项重要技术。基质辅助激光解吸电离飞行时间（Matrix Assisted Laser Desorption/Ionization Time Of Flight，MALDI-TOF）质谱仪与标准生化分析相结合，可以用于测定生物大分子的分子量以及主要的序列信息（Kim 等人，2004）。

这种飞行时间质谱仪的基本工作原理是：特定时空下的一群离子在恒定的电场作用下飞过电场管道时，离子穿越管道的飞行时间与离子的质荷比（m/z，即带电粒子的质量与所带电荷之比）成正比。

MALDI-TOF 质谱仪的谱分析存在一个问题，就是分子量较小的复合物产生的信号与大分子碎裂片段信号之间会产生混淆。因此，要完成准确的谱分析，就要采用算法消除碎裂分子片段引起的基线偏差。

请编写一个 MATLAB 程序，用于消除 MALDI-TOF 质谱中存在的碎裂分子片

⊖ 原文此处为规则 4。—译者注

⊜ 原文此处为规则 2、3 和 4。—译者注

段引起的基线偏差。

解：

碎裂片段引起的质谱变化比较缓慢，不形成非常陡峭的尖峰。并且，质荷比（m/z）采样点完全由被测试的分子及其所用的基质决定，其采样间隔可能有所变化。因此，必须进行两步处理：

1）重新等距抽取质谱信号，即降低采样频率；2）消除质谱信号中较低频率的碎裂片段成分。

在 MATLAB 中，利用样条函数拟合原始采样数据，可以方便地完成这两步处理操作。质谱的重采样就是将质荷比（m/z）数据点进行等间隔抽样；碎裂片段的影响则可以通过减去基线分量来消除。下面是用于数据处理和显示的 MATLAB 程序。

MALDI-TOF 质谱数据存于 clear. xls，该文件可以在本书的网站上下载。文件由两列数据组成，第一列为质荷比，第二列为质谱的频率计数值。图 9. 11 为该数据的图示，横坐标为质荷比，纵坐标为频率计数值。图中共有 16 727 个采样数据，在 4817. 2 处有一个尖峰，是由实验中使用的底物引起的。读取文件并显示数据的程序为

```
data = xlsread('clear');
figure(1);
plot(data(:,1),data(:,2))
```

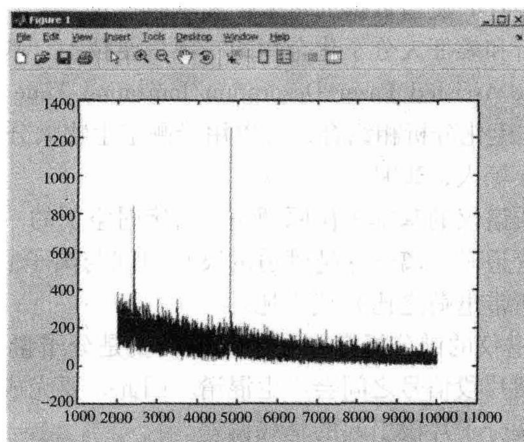


图 9. 11　MALDI-TOF 质谱数据文件 clear. xls 的数据显示

下面先取质谱的前 200 个采样数据进行处理，其质荷比（m/z）的范围为 2000 到 2060。图 9. 12 的显示程序如下：

```
% work with the first 200samples
mz = data(1:200,1);
y = data(1:200,2);
figure(2);
plot(mz,y);
```

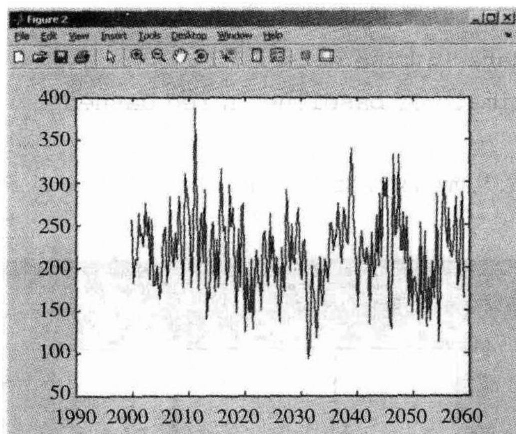


图 9.12 clear.xls 文件中的前 200 个采样数据

然后对这段数据重新抽样，从 200 个数据中抽出 100 个数据用于样条拟合，也就是样条多项式要恰好经过这 100 个数据，而另外 100 个数据则由估计值求得。程序如下：

```
% decimate the MALDI data to estimate the baseline
mz_dec = mz(1:2:200);
y_dec = y(1:2:200);
% compute the spline estimate from the decimated data
% interpolate on the orginal m/z range
% pp is the piecewise polynomial
% force the endpoint conditions to be zero slope
% as described in the textbook
y_dec = [y(1);y(1:2:200);y(200)]
pp = spline(mz_dec,y_dec);
```

MATLAB 函数 spline 返回的值 pp 是分段样条多项式，变量 mz_dec 存放的是重抽样后的 100 个质荷比的值，变量 y_dec 存放的是对应于这些质荷比的频

率计数。

下面利用已求得的样条多项式，插值估计其他 100 个数据点的值，重新达到 200 个数据点（见图 9.13）。程序如下：

```
% evaluate the polynomial on the original m/z range
yy=ppval(pp,mz);
% plot both:
% original data in blue dots
% spline estimate of baseline in red dashes
figure(3)
plot(mz,y,'b.',mz ,yy,'r - - ')
```

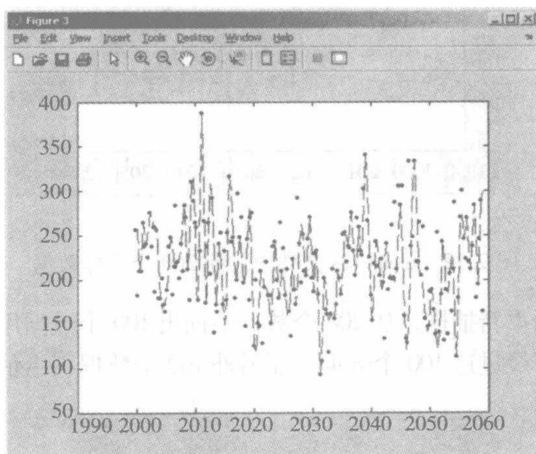


图 9.13 clear. xls 文件数据的插值基线

图 9.13 将原始质谱数据和插值后的数据显示在一起，其中的圆点表示原始质谱数据，虚线表示插值基线。下面将原始数据减去插值基线就可以去除基线。

```
% do the baseline removal:
% subtract the spline estimate from the raw data
figure(4)
plot (mz,y-yy)
```

注意，图 9.14 的基线为 0，剩余的频率计数值就代表了测试样品分子产生的分量。

将此处理过程稍作改动，用少于 50% 的数据估计基线，就可以将此处理过

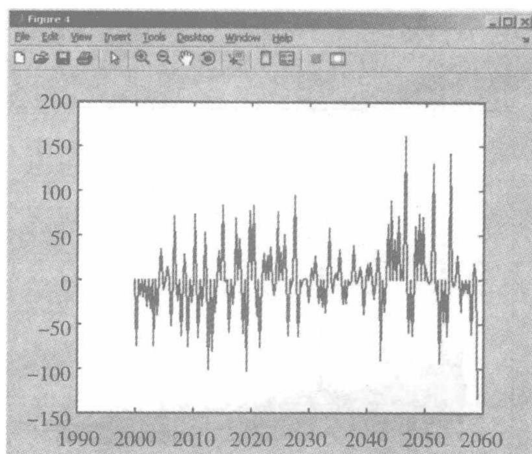


图 9.14　去除基线之后的样品数据

程应用于整个质谱数据。如下例子中，用少于 10% 的数据估计基线，这些数据的线性分布抽取点用 MATLAB 的 linspace 函数计算，并用 fix 函数将下标值取为整数。程序如下：

```
mz = data(:,1);
y = data(:,2);
indicies = floor (linspace(1,length(mz),1000));
mz _ dec = mz(indicies);
y _ dec = [y(1);y(indicies);y(length(mz))];
pp = spline(mz _ dec,y _ dec);
yy = ppval (pp,mz);
figure(5)
plot(mz,y,'b.',mz,yy,'r - - ')
figure(6)
plot(mz,y - yy)
```

图 9.15 所示的虚线表示基线变化，将此基线减去，得到的质谱数据如图 9.16 所示，可见，4817.2 处的尖峰仍然存在，但是，碎裂片段引起的基线分量已被去除。

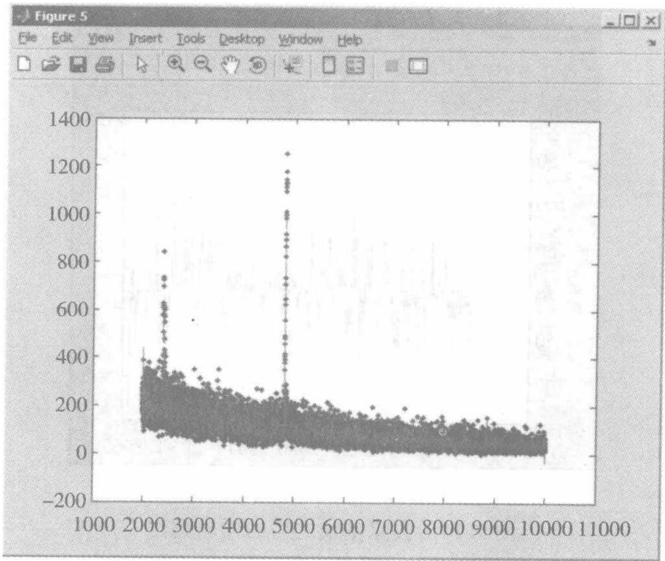


图 9.15　用少于 10% 的数据估计的基线变化

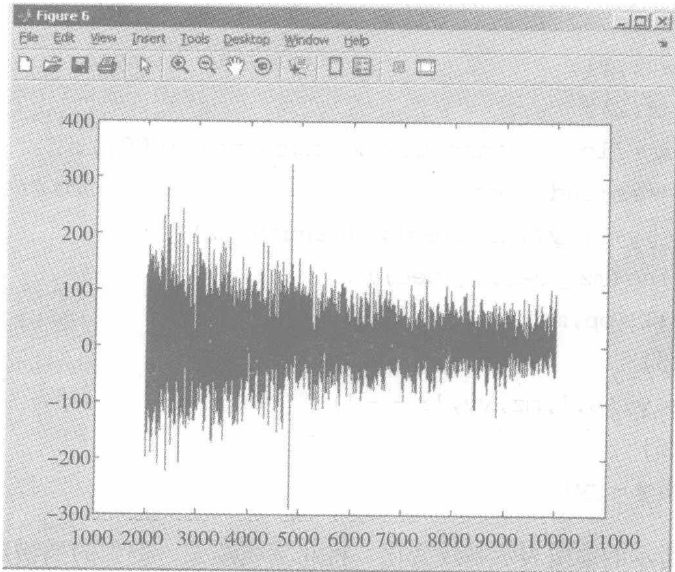


图 9.16　去除图 9.15 所示的基线分量之后的 MALDI-TOF 数据

9.7 傅里叶变换

时频分析是生物医学信号处理的重要方法之一，MATLAB 有这方面的基本工具函数。本书仅结合这些函数指令介绍一些基本概念，不深入讲解时频分析的详细内容，第 10 章给出了一些相关的例题。有关时频分析的深入内容，请读者参考 Bracewell（1999）和 James（2002）等人的著作。

傅里叶变换是一种特殊的曲线拟合，就是用周期函数来拟合数据。

它将某个自变量的函数转换成另一个自变量的函数。通常，利用傅里叶变换将时间函数的数据转变为周期函数的数据，这些周期函数用正弦波表示。

因此，用傅里叶变换转换信号时，两个自变量就分别是时间和频率。大家已经知道，连续变量和离散变量之间是有区别的，因此，连续的和离散的时间（或频率）也是不同的，于是，就有了不同形式的傅里叶变换，这些变换可以归纳如下：

		频率	
		连续	离散
时间	连续	傅里叶变换	傅里叶级数
	离散	离散傅里叶变换	有限傅里叶变换

系统分析课程中所介绍的用于理论分析的傅里叶变换为

$$F(\omega) = \int f(t)\mathrm{e}^{-\mathrm{j}\omega t}\mathrm{d}t \tag{9.51}$$

它是用正弦波周期信号表示时间函数。积分符号里面是两个连续函数$f(t)$ 和 $\mathrm{e}^{-\mathrm{j}\omega t}$ 的乘积，$\mathrm{e}^{-\mathrm{j}\omega t}$是正弦波 sin（$\omega$）和 cos（$\omega$）的指数表达形式。任意两个函数的乘积其实是两函数之间相似程度的测量，对乘积再求积分得到的系数 $F(\omega)$ 就是自变量取值范围内这种相似性的总和。也就是，系数 $F(\omega)$ 是频率ω在函数$f(t)$ 中含量大小的测量。

傅里叶反变换为

$$f(t) = \frac{1}{2\pi}\int F(\omega)\mathrm{e}^{\mathrm{j}\omega t}\mathrm{d}\omega \tag{9.52}$$

其含义与傅里叶变换相似。在积分符号里面也是两个函数的乘积，即 $F(\omega)$ $\mathrm{e}^{\mathrm{j}\omega t}$。此积分也是相似性的总和，表示频率 ω 在时间函数$f(t)$ 中所占的份量。

这两个变换形成了傅里叶变换对，可以将信号变换到较合适的变量域中进行分析。但是，该变换对中的自变量和因变量都是连续的，而 MATLAB 等计算环境只能用变换的离散形式。

相应的离散傅里叶变换的两个变量都是离散的采样信号，连续积分也变成了

离散的加和运算。其中，连续变量 t 变成了一系列离散时间点 $n\tau$，n 为下标，也就是采样数，τ 为两个采样点之间的时间间隔。习惯上，第一个采样点的下标设为 $n=0$，最后一个采样点的下标设为 $n=N-1$。

奈奎斯特定理说明了要用怎样的信号采样密集度，才能准确表达原始信号。根据奈奎斯特定理，如果信号的最高频率成分为 ω_N，那么，信号的采样速率至少为 τ：

$$\omega_N \leqslant \frac{2\pi}{\tau} \tag{9.53}$$

这个定理经常被解释为：如果信号以 τ 的速率采样，那么可以测量的信号的最高频率为 ω_N。

离散傅里叶变换用连续频率项表示采样数据函数，采样数据是离散的，因此，时间上的积分必须用采样数据的离散求和代替，但频率还是连续的，即

$$F(\omega) = \sum f(n\tau)\mathrm{e}^{-\mathrm{j}\omega n\tau} \tag{9.54}$$

从连续频率到离散时间的反变换还是积分：

$$f(n\tau) = \frac{\tau}{2\pi}\int F(\omega)\mathrm{e}^{\mathrm{j}\omega n\tau}\mathrm{d}\omega \tag{9.55}$$

但是，其积分限被定为 2π 范围。分析奈奎斯特定理就可以理解这个积分限的由来。我们已经知道，对于采样速率 τ 可以测量的最高信号频率为 ω_N，因此，这意味着采样数据中不包含高于 ω_N 的频率成分，也不会有低于 0 的频率成分。所以，可以限定该积分限。并且，$0\sim\omega_N$ 的积分限等价于 $(-\omega_N/2)\sim(\omega_N/2)$ 的积分限。利用 $\omega=2\pi$ 这个关系，已知采样频率 $f_s=1/\tau$，反变换的积分限就是 $(-\pi/\tau)\sim(\pi/\tau)$。

如果将频率离散化，就是另一回事了。离散频率由基波频率和谐波频率组成，谐波频率有一组频率，其最高频率是奈奎斯特频率。傅里叶级数就是用基频波及其谐波来表示函数 $f(t)$，其时域到频域的变换为

$$F_n = \frac{1}{T}\int_0^T f(t)\mathrm{e}^{-\mathrm{j}\omega_0 nt}\mathrm{d}t \tag{9.56}$$

由频域到时域的反变换则为

$$f(t) = \sum_{n=-\infty}^{\infty} F_n \mathrm{e}^{\mathrm{j}\omega_0 nt} \tag{9.57}$$

其中，$\omega_0=2\pi f_0$ 为基频。下标 n 是基频的倍数，即谐波频率。可见，这种 $f(t)$ 的傅里叶级数表达假设信号在 $n\omega_0$ 与 $(1+n)\omega_0$ 之间不存在频率成分，并且，实际上相当于把信号看成是原信号以积分限 $0\sim T$ 为周期重复的周期性信号。

有限傅里叶变换将离散的时间函数映射到离散的频率域。其变换公式为

$$F_u = \sum_{n=0}^{N-1} f(n\tau)\mathrm{e}^{-\mathrm{j}\omega_0 nu} \tag{9.58}$$

反变换公式为

$$f(n\tau) = \sum_{u=0}^{N-1} F_u \mathrm{e}^{\mathrm{j}\omega_0 nu} \tag{9.59}$$

但是，此处的离散频率与傅里叶级数的离散频率不同，它们以离散频率序列的长度 N 为周期重复。这里的时间序列和频率序列都周期性重复，时间上的周期是采样信号的长度，频率上的周期则是离散频率求和的范围。

首字母缩写 FFT 通常指快速傅里叶变换，是有限傅里叶变换的一种有效算法。MATLAB 的 fft 和 ifft 指令分别用于计算快速傅里叶变换及其反变换。MATLAB 文档把这两个指令称为离散傅里叶变换的函数，这不太确切，应该是有限傅里叶变换。

例 9.9　脑电图（EEG）信号的频率成分分析。

脑电信号是在头皮上记录到的神经电活动信号，EEG 的某些变化是大脑对外界事件的响应，因此，这些电位变化被称为事件相关电位（Event - Related Potentials，ERP）。

EEG 信号及其分析是了解大脑生理基础以及思维活动过程的工具。EEG，尤其是 ERP，是研究各种思维任务处理以及各种行为状态与大脑活动区域之间对应关系的有用方法。

某些行为状态与特定的 EEG 频率成分相关，例如：

1）0 ~ 2 Hz 的 δ 波是 1 岁以下婴儿脑电的主要节律，也是成人睡眠进入第 3 和第 4 阶段时脑电的主要节律。

2）2 ~ 7 Hz 的 θ 波称为慢波，如果出现在清醒的成人脑电中是不正常的，但是，在 13 岁以下的儿童脑电中完全正常，出现在睡眠期也是正常的。

3）在闭目和放松状态下，脑电中会出现 7 ~ 13 Hz 的 α 波，睁眼或者进入思考、计算等警觉状态后，α 波即消失。α 波是正常成人（一般在 13 岁以后）处于休息状态时的主要节律。

4）从 14 ~ 64 Hz 的 β 波通常认为属于正常脑电信号。但 β 波也是某些病人脑电的主要节律，如处于戒备或抑郁状态的病人。

请编写一个 MATLAB 程序，将 EEG 信号的 δ、θ、α 和 β 这几种波的频率成分分开，并计算各个频率段的能量。

解：

图 9.17 所示是右侧中脑上方记录的 EEG 信号时长 10s，采样频率为 250Hz，共有数据点 2500 个，幅值范围为 - 20 ~ 20 mV。数据来源于 Garrett 等人的工作（2003）。

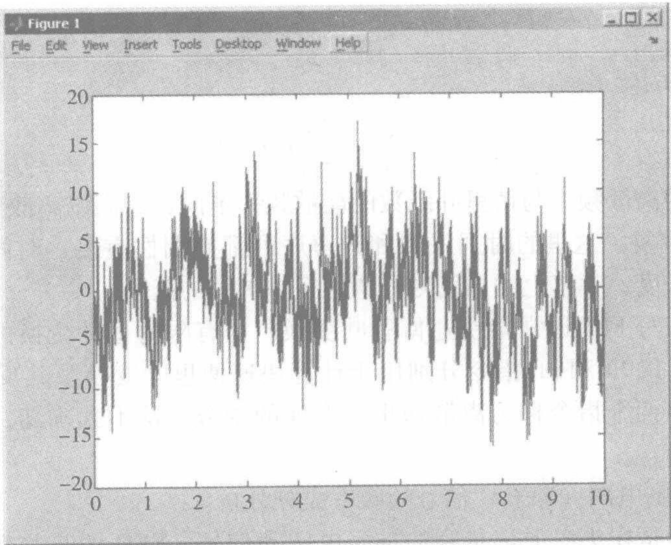


图 9.17 采样频率为 250 Hz 的 10sEEG 信号

假设 EEG 数据存放在变量 eegchannel 中，如下 MATLAB 指令可以计算 EEG 的有限傅里叶变换并作图显示结果：

```
eegfreq = abs(fft(eegchannel));
plot(eegfreq);
```

fft 指令的输出是一个复数矢量，包含了频率成分的实部和虚部，再用 MATLAB 的 abs 函数可以求得这个复数矢量的大小，也就是图 9.18 所示的频谱信号。

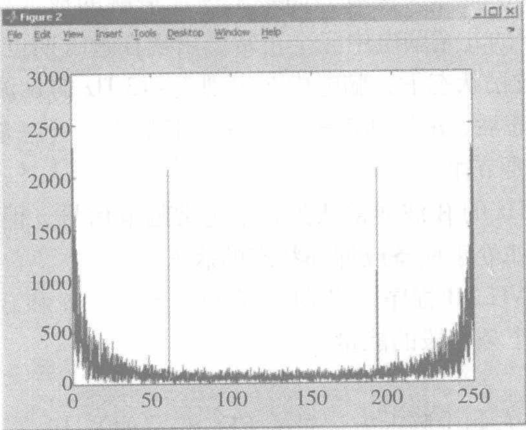


图 9.18 EEG 的频谱信号

如上所述，快速傅里叶变换在时域和频域上都是周期信号，即频谱信号应该

是对称且重复的。另外，根据奈奎斯特准则，采样信号可以包含的最大频率只有采样频率的一半，因此，此例的最大频率为 125 Hz。从图 9.18 可见，125 ~ 250 Hz 的频谱实际上与 -125 ~ 0 Hz 的频谱相同。

利用 MATLAB 的另一条指令 fftshift，可以重新显示频域数据，把 0 Hz 的直流（DC）分量放在图的中间，也就相当于把奈奎斯特频率（125 Hz）以上的正频率信号用 -125 ~ 0 Hz 的负频率段信号代替了。指令如下：

```
eegfreqshift = fftshift(eegfreq);
plot(eegfreqshift);
```

输出结果如图 9.19 所示，整个频谱的显示范围为 -125 ~ 125 Hz，中心位于 0 Hz。两个对称的尖峰对应于 ±60 Hz，是交流电源的噪声引起的。在进行下一步分析之前，要去除直流分量和 60 Hz 分量。

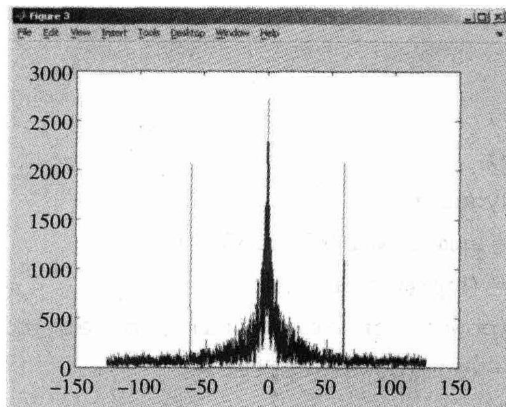


图 9.19　中心位于 0 Hz 的频谱

去除 0 Hz 和 60 Hz 分量的最简单方法就是把频谱中这些频率上的值置 0，但这并不是好方法。程序如下：

```
% remove DC component
eegfreqshift(1240:1260) =0;
% remove 60Hz
eegfreqshift (645:655) =0;
eegfreqshift(1845:1855) =0;
```

于是，频谱就变成了图 9.20 所示的形态。

各频率段信号所占能量的比率就是各频率段信号能量除以信号总能量，计算如下：

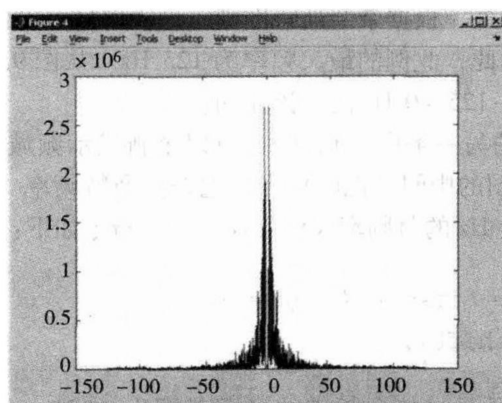


图 9.20 去除直流和 60 Hz 分量之后的频谱（其中心位于 0 Hz）

```
power = eegfreqshift.^2;
delta = 1:20;
theta = 20:70;
alpha = 70:130;
beta = 140:640;
center = 10 * Nyquist;
power_total = sum(power(650:1850));
power_delta = (sum(power(center -
delta)) + sum(power(center + delta)))/power_total
power_theta = (sum(power(center -
theta)) + sum(power(center + theta)))/power_total
power_alpha = (sum(power(center -
alpha)) + sum(power(center + alpha)))/power_total
power_beta = (sum(power(center -
beta)) + sum(power(center + beta)))/power_total
```

程序运行之后输出的各频率段信号能量的比率为

```
power_delta =
     0.2453
power_theta =
     0.3883
power_alpha =
     0.1690
```

```
power _ beta =
      0.2125
```

可见，2～7 Hz的θ频率为主要成分，这在非清醒状态的成人脑电中属于正常现象。

9.8 本章学习要点

学完本章之后，读者应该掌握以下内容：

1）任何测量都有变化，有些是由测量误差引起的，有些则是由生物变异性引起的；

2）描述性统计量用于表征测量数据的样本；

3）统计推断用于从样本数据中总结和归纳总体参数并得出结论；

4）最小二乘法是用数学模型拟合测量数据的方法；

5）插值多项式是用数学模型拟合准确数据；

6）4种傅里叶变换是用周期性数学模型拟合数据，使用的是正弦波；

7）MATLAB有内置函数可以用于计算描述性统计量、统计推断、插值样条，以及傅里叶变换。最小二乘法则可以方便地利用矩阵矢量的算术运算来完成。

9.9 习题

9.1 可视化人体计划（Visible Human Project，VHP）的网站（http：//www. nlm. nih. gov/research/visible/visible _ human. html）上新鲜尸体和冰冻尸体的CT图像都有。请选择其中的一对图像，比如头部或胸部的图像，分别计算两幅图像的描述性统计量，并且分析两幅图像亮度分布的差别。这些差别是否反映了新鲜组织和冰冻组织之间的区别？

9.2 从VHP网站的CT数据库中选择一幅足部的图像，编写一个MATLAB脚本，读取该图像数据，显示图像，并将图像进行阈值处理，使趾骨为白色，肌肉和肌腱等软组织为黑色。

9.3 生理网（PhysioNet）是由美国国家卫生研究院（NIH）下属的国家研究资源中心（National Center for Research Resources，NCRR）维护的生理信号数据库。网址为：http：//physionet. incor. usp. br/。该数据库中包含有文本格式的心电图（ECG）记录信号的样本，下载网址为：http：//www. physionet. org/cgi-bin/rdsamp。请从生理网的MIT-BIH数据库中下载一段正常窦性心率（MIT-BIH Normal Sinus Rhythm）的心电图记录样本，编写一个MATLAB脚本程序，去除心电图信号中的60 Hz噪声（提示：请参考例题9.9的方法）。

9.4 将上题中编写的MATLAB脚本程序用于处理MIT-BIH数据库中一段心律失常（Arrhythmia）的心电图信号样本，并考察心律失常的ECG中所包含的频率成分与正常窦性心率的ECG是否有所不同。

9.5 MIMIC数据库（Moody和Mark，1996）也是生理网上的一部分，它保存的是多参数监护

仪记录的信号。请下载其中的第055号记录信号，编写一个MATLAB脚本程序，用9.4节介绍的相关系数分析记录中的任意一对信号是否相关，例如，动态血压（Ambulatory Blood Pressure，ABP）波形与气道正压通气（Positive Airway Pressure，PAP）波形之间是否显著相关？呼吸压（RESP）与气道正压通气PAP之间是否显著相关？

9.10 参考文献

Bracewell, R. N. 1999. The Fourier Transform & Its Applications. 3rd ed. New York: McGraw-Hill Book Company.

Garrett, D., Peterson, D. A., Anderson, C. W., and Thaut, M. H. 2003. Comparison of Linear and Nonlinear Methods for EEG Signal Classification. *IEEE Transactions on Neural Systems and Rehabilitative Engineering*, **11**(2):141-144.

Goldberger, A. L., Amaral, L. A. N., Glass L., Hausdorff, J. M., Ivanov, P. Ch., Mark, R. G., Mietus, J. E., Moody, G. B., Peng, C. K., and Stanley, H. E. 2000. PhysioBank, PhysioToolkit, and Physionet: Components of a New Research Resource for Complex Physiologic Signals. *CIRCULATION*, **101**(23):e215-e220.

James J. F. 2002. A Student's Guide to Fourier Transforms With Applications in Physics and Engineering. Cambridge: Cambridge University Press.

Kim, S., Ulz, M. E., Nguyen, T., Li, C., Tycko, B., and Ju, J. 2004. Digital high-throughput multiplex genotyping in human tumors. *Genomics*, **83**:924-931.

Moody, G. B., and Mark, R. G. 1996. A Database to Support Development and Evaluation of Intelligent Intensive Care Monitoring. *Computers in Cardiology*, **23**:657-660.

相关网址：

可视化人体计划（Visible Human Project）：

http：//www. nlm. nih. gov/research/visible/visible _ human. html

美国国家医学图书馆（NLM）：

http：//www. nlm. nih. gov/

美国国家卫生研究院基因数据库（NIH Gene Expression）：

http：//www. ncbi. nlm. nih. gov/genome/guide/human/resources. shtml

GSE28 基因：

http：//www. ncbi. nlm. nih. gov/entrez/query. fcgi？ db = gds&cmd = search&term = GSE28

第 10 章 生物医学系统的建模仿真:应用举例

10.1 生物医学系统的数学模型

本章将举例说明数值方法以及 MATLAB 和 Simulink 等计算工具在生物医学工程领域中的应用。每个例题都首先陈述临床问题或者生理问题，再介绍建模方法，然后说明如何用 MATLAB 和/或 Simulink 建立数值求解的计算机算法。

其中有些例子来自公开的免费资源，例如，10.2 节介绍的生理网（PhysioNet）。该网站有一个很大的生物电信号数据库，也有一些用于分析研究的软件工具。这里所举的例题简单说明了生理网的使用方法，10.3 节使用的脑电图（EEG）信号，以及 10.7 节的 Simulink 仿真软件 PHYSBE 都来自生理网。本章中的其他例题则取自有关生物医学工程的已发表文献。

阅读学习本章的例题不需要按顺序进行，表 10.1 只是根据研究领域以及所使用的数值方法将所有的例题进行了分类和归纳，并标出了使用 Simulink 的那些例题。

表 10.1 本章应用例题的小结

章节	例题	应用	研究领域	模型和算法
10.2	10.1、10.2	PhysioNet 和 PhysioBank	生物电	模型拟合 常微分方程
10.3	10.3	大脑活动	生物电	模型拟合 统计学
10.4	10.4	糖尿病及其胰岛素治疗	生物化学	常微分方程 Simulink
10.5	10.5	肾清除率	生物化学	常微分方程
10.6	10.6	步态以及运动分析	生物力学	数值线性代数
10.7	10.7 ~ 10.12	PHYSBE	生物化学	常微分方程 Simulink

本章主要包括如下学习内容:

1) 学习如何应用 MATLAB 求解生物医学工程问题;

2)·学习如何应用 Simulink 仿真生理系统模型;

3) 编写 MATLAB 程序求解生物医学工程问题;

4) 开发 Simulink 生理系统仿真模型;

5) 认识数值方法在数学模型与计算机求解之间所起的桥梁作用。

10.2 PhysioNet、PhysioBank 和 PhysioToolkit

生理网(PhysioNet)是一个用于生物医学研究和开发的互联网资源,由美国国家卫生研究院(National Institute of Health)下属的国家研究资源中心(National Center for Research Resources)主办。生理网中的生理库 PhysioBank 在第 9 章已介绍过,其中包含有研究人员提供的生理信号和标准化的生理信号及其注释文档。第 9 章使用了 PhysioBank 的几个样本,并举例说明了如何利用模型解答生理问题。

生理工具箱 PhysioToolkit 是生理网的另一个组成部分,它是一个软件库,可用于分析生理库的数据,使数据可视化,进行信号处理、软件开发、以及仿真运算。其中某些软件工具可以由 MATLAB 调用,下面将举例说明。

10.2.1 ECG 仿真

第 2 章曾经用 MATLAB 脚本产生过一个非常粗略的心电图 QRS 复合波的仿真曲线。PhysioToolkit 有一个用于生成正常 QRS 波和心律失常 QRS 波的 MATLAB 脚本,名为 ECGwaveGen,由 Floyd Harriott 提供,基于 Ruha 等人 1997 年的论文编写。该 MATLAB 脚本允许用户改变心率、信号长度、采样频率、QRS 波幅值和持续时间以及 T 波幅值等仿真信号的参数。下面的例题说明了其使用方法。

例 10.1 用 MATLAB 脚本 ECGwaveGen 产生 ECG 波形。

请从网址 http://www.physionet.org/physiotools/matlab/ECGwaveGen/下载 ECGwaveGen.m 和 QRSpulse.m 两个 MATLAB 脚本文件,并用这两个程序分别生成 5s 长的正常心率 ECG 波形和 5s 长的窦性心动过速 ECG 波形。

解:

正常心率有时也称为窦性心率,使用 ECGwaveGen 的默认参数就可以生成。ECGwaveGen 的帮助信息如下:

```
[QRSwave] = ECGwaveGen(bpm, dur, fs, amp) generates an artificial
```

```
ECG/EKG waveform
    Heart rate (bpm) sets the qrs event frequency (RR interval).
    Duration of the entire waveform (dur) is in units of seconds.
    Sample frequency (fs) sets the sample frequency in Hertz.
    Amplitude (amp) of the QRS event is measured in micro Volts. The
waveform consists of a QRS complex and a T-wave. No attempt to repre-
sent a P-wave has been made.

    There are two additional parameters that can be changed from
within the function. They are the parameters that set the QRS width
(default 0.1 secs) and the t-wave amplitude (default 500 uV).
```

MATLAB 的指令为

```
Norm = ECGwaveGen(60,5,64,1000);
time = [(1:length(Norm))/64];
plot(time,Norm);
```

这些指令用于生成 QRS 复合波，建立一个时间矢量，并如图 10.1 所示，以时间为横坐标，复合波幅值为纵坐标，作出 ECG 波形图。其中，QRS 复合波峰值约为 0.75 mV，T 波峰值为 0.5 mV。

窦性心动过速是指心率超过每分钟 100 次，即 100 beat/min（简称 bpm）。以下的 MATLAB 程序仿真 110 bpm 的窦性心动过速 ECG 波形，其输出结果如图 10.2 所示。

```
figure(2);
Norm = ECGwaveGen(110,5,64,1000);
time = [(1:length(Norm))/64];
plot(time,Norm);
title('ECGwaveGen synthesis of Sinus Tachycardia');
xlabel('Time in seconds');
ylabel('Amplitude in microvolts');
```

PhysioNet 提供的 ECGwaveGen 脚本不能产生心率大于 110bpm 的窦性心率过速 ECG 波形。该网址上还有另一个名为 ecgsyn 的脚本，基于 McSharry 等人 2003 年发表的论文编写。ecgsyn 产生的 ECG 波形在形状和频率上都可以有更大的变化，它用 3 个联立微分方程仿真波形，有关微分方程组求解的详细内容请参阅第

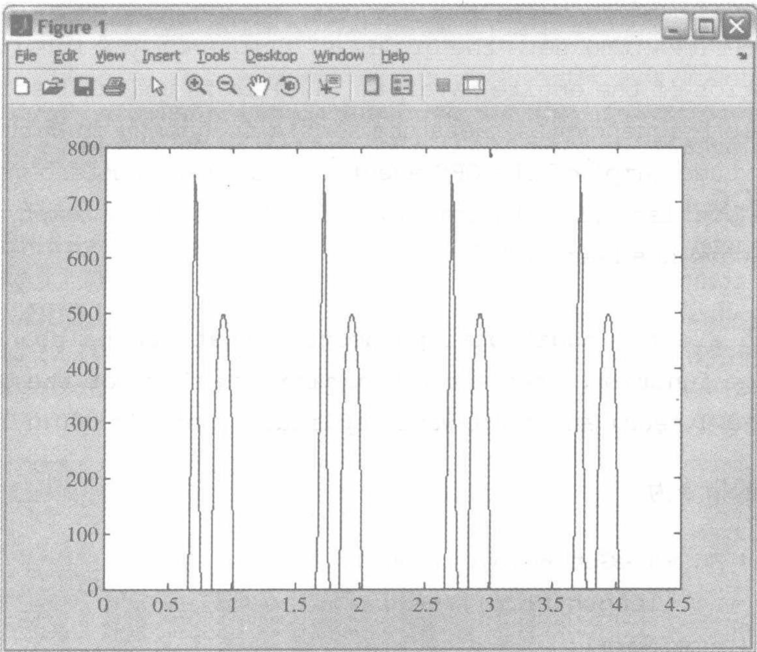


图 10.1　ECG 波形

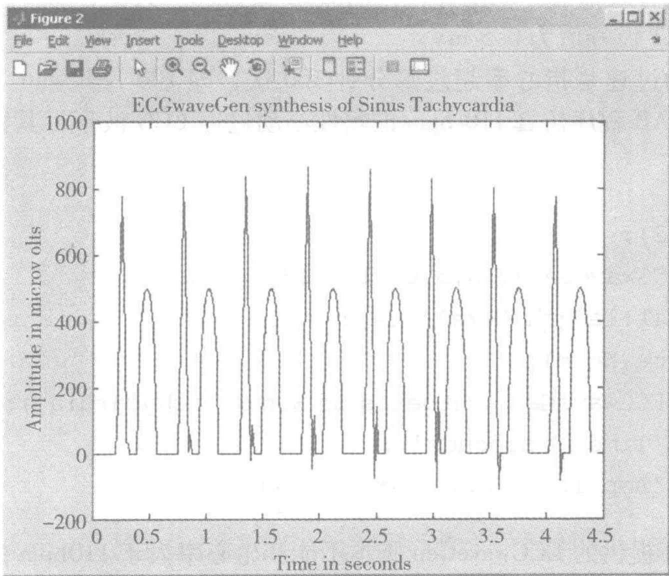


图 10.2　窦性心率过速的仿真波形

7 章。该 MATLAB 脚本 ecqsyn. m 的下载网址为 http：//www. physionet. org/physiotools/ecgsvn/。

如果不设定任何参数，ecqsyn 脚本产生的 ECG 复合波具有如下默认参数：

ECG 采样频率：256Hz；

ECG 复合波数目：约 256 个；

测量噪声幅值：0；

平均心率：60bpm；

心率的标准差：1bpm；

低频与高频的比率（LF/HF）：0. 5；

内部采样频率：512Hz。

如果要生成不含噪声的具有正常心率的 10 个 ECG 波形，且采样频率为 64Hz，则 MATLAB 指令为：

```
s=ecgsyn (64,10, 0, 60) ;
```

采样频率为 64Hz，因此，用以下指令显示的前 256 个采样点包含如图 10. 3 所示的 4 个 ECG 复合波。

```
plot(s(1:256))
```

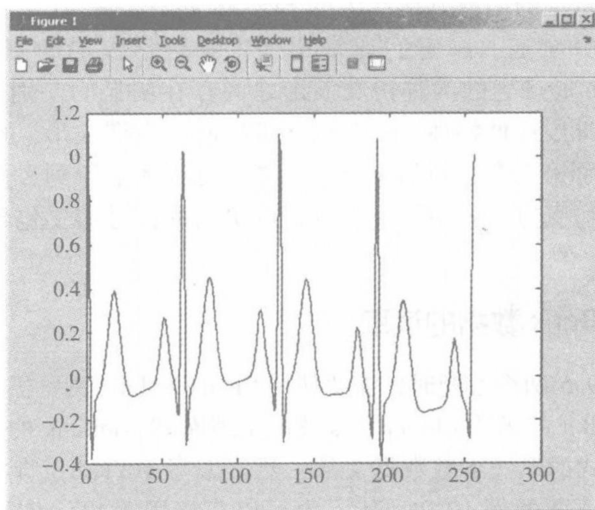


图 10. 3　利用 ecgsyn 产生的正常心率 ECG 波形

ecgsyn 调用中的第 4 个参数表示心率。如果要仿真窦性心动过速，可以用以下指令：

```
s=ecgsyn(64, 16, 0, 110);
plot(s(1:256))
```

此时显示的256个采样数据如图10.4所示，图中相同时间段上PQRST复合波的数目大约是图10.3的两倍。

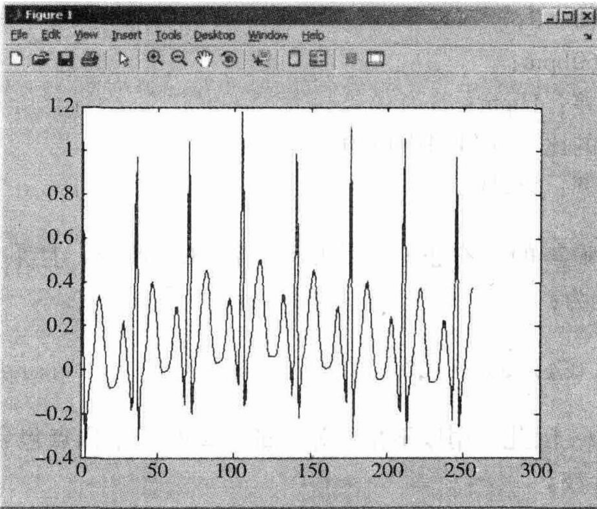


图10.4 利用ecgsyn产生的窦性心动过速ECG波形

显然，利用特定的函数方程产生的ECG波形与实际ECG波形具有一定的相似性，但并不完全逼真，它们通常用于生物医学仪器的测试。上述ECG-waveGen和ecgsyn所产生的两种仿真波形之间存在明显的差别，ecgsyn的输出包含了P波和S-T段压低特性，因此更接近实际心电图波形。可见，较逼真的波形仿真需要求解第7章介绍的3个联立微分方程才能得到，而第9章所用的基于曲线拟合的仿真方法就不行。逼真程度更高的信号需要包含更多真实信号的频率成分。

10.2.2 PhysioBank数据的读取

求解9.3和9.5两个习题时，需要使用Chart-O-Matic浏览器，通过调用生理工具箱PhysioToolkit的函数rdsamp，来读取生理库PhysioBank的数据文档。用户首先要以文本格式的形式下载数据文件，然后编写MATLAB脚本读取该文本文件。实际上，如下面的例10.2所示，也可以直接用MATLAB脚本读取Physio-Bank的原始数据文档。

例10.2 读取PhysioBank数据及其注释，并作图显示信号。

生理库PhysioBank中的ECG数据用一种称为“212”的特殊格式保存。请

下载习题 9.3 所用的样本文件，直接用 MATLAB 脚本 rddata.m 显示数据。

解：

我们在前面强调过改变 MATLAB 函数的参数值可以获得不同的调用结果。但是，rddata.m 脚本不是通过函数参数，而是用赋值语句来设置路径名、头文件名、属性文件名和数据文件名等参数，因此，要改变这些参数，就需要修改这些赋值语句。如果用 MIT-BIH 样本数据（共包含 3 个文件：头文件 100.hea、属性文件 100.atr 和数据文件 100.dat），则运行 rddata.m 的输出结果如图 10.5 所示。

这个 ECG 记录信号的数据总长有 80 多秒，粗略一看，很难分辨图中显示的波形。可以从 MATLAB 工作空间存放的变量中截取小段数据显示，其中 MATLAB 变量 *M* 存放两路 ECG 采样信号，每路信号占数组的一列。

采样频率存放在另一个变量 sfreq 中，该样本的采样频率为 360Hz。如果心率为 60 beat/min，则在该采样频率下，4 个 QRS 复合波约占 1440 个采样点，用以下指令显示记录文件的前 1440 个采样数据，结果如图 10.6 所示。

```
plot(M(1:1440, 1))
```

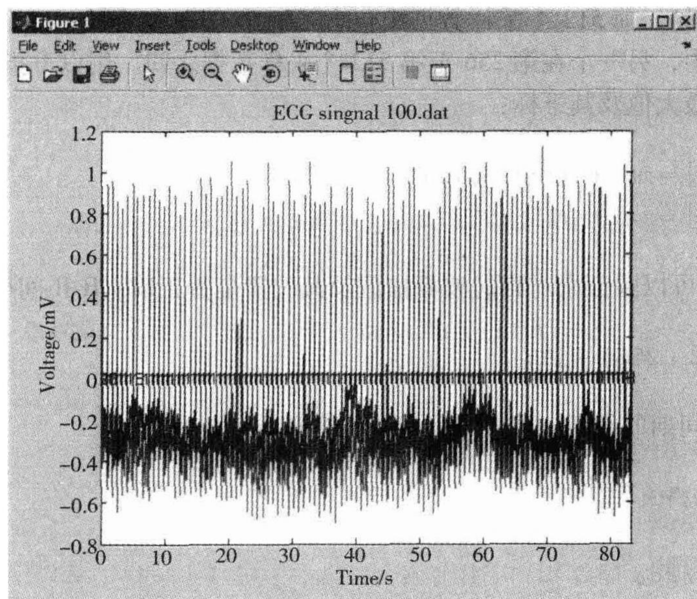


图 10.5 MIT-BIH 数据库的 ECG 样本数据

由图 10.6 可见，该记录样本的实际心率大于 60bpm。利用存放在 MATLAB 变量中的数据，通过寻找 R 波的波峰，计算 R-R 间期，很容易求得实际心率。

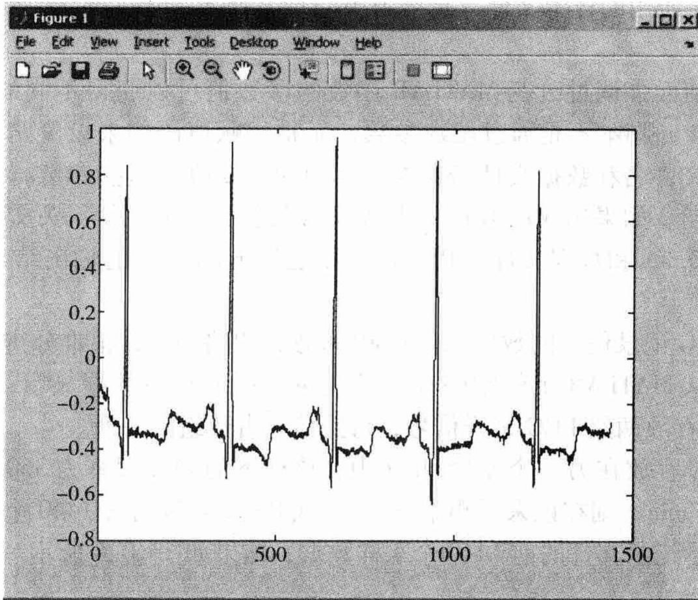


图 10.6 MIT-BIH 数据库 100.dat 数据文件的前 1440 个采样数据

由图 10.6 可见，前 512 个采样数据中包含了两个 QRS 复合波，一个位于前 256 个采样数据中，另一个在第 256 至第 512 个采样数据之间。用以下指令可以找出这两个波的最大值及其下标：

```
[p1,i1] =max(M(1:256,1)) ;
[p2,i2] =max(M(256:512,1)) ;
```

对照图 10.6 可以验证所求得的波峰幅值及其位置是否正确。R-R 间期则为

```
RR = (i2 +256) - i1;
```

心率为 R-R 间期的倒数，已知采样频率为 360Hz，心率则为

```
360 * 60/RR
ans =
    73.4694
```

本例题说明了读取 PhysioBank 原始数据的方法，有了这个基础，就可以应用本书所介绍的各种数据处理和分析方法来研究这些生理信号的特性。

10.3　信号处理——EEG 数据分析

通过脑电图（EEG）信号的分析，可以了解大脑生理和思维行为的动态过程；也可以确定与特定任务处理和行为状态相对应的大脑活动区域。本节下面所举的例子是要测定在执行某种思维任务时大脑左右两个半球的活动是否有差别。

记录 EEG 信号所采用的标准电极放置法称为 10-20 系统。如图 10.7 所示，名称中的“10”和“20”是指从鼻根经过头顶到枕外粗隆的联结线分段以后，各段占总弧长的百分数。

位于左右耳垂的 A1 和 A2 作为参考电极，它们也是眼动电位（Electro-OculoGram，EOG）的记录电极，EOG 记录的是 EEG 信号中由于眨眼引起的肌电干扰信号。

下面所述研究项目的数据取自 Keirn（1988）的硕士论文工作，由科罗拉多州立大学（Colorado state university）计算机系的 Charles Anderson 教授提供。这些数据中除了 EOG 信号以外，还有 O1、O2、P3、P4 以及 C3、C4 电极记录的信号。

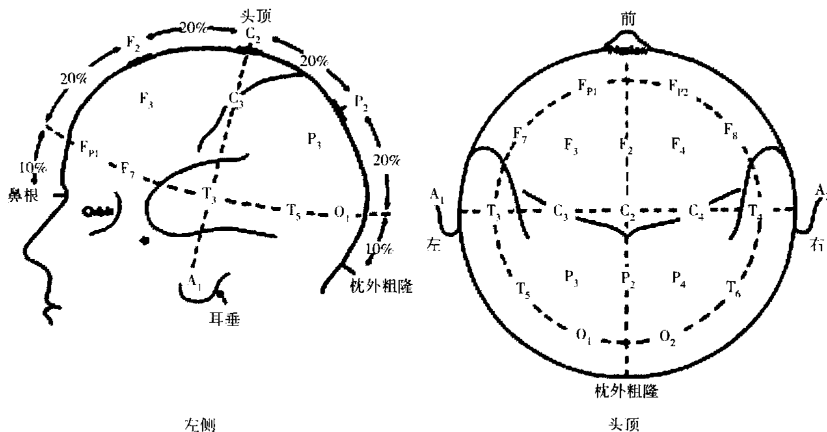


图 10.7　EEG 记录的国际 10-20 系统电极放置法（摘自 Jasper 1958）

Keirn 的实验研究中，要求测试对象执行下列 5 项任务：

1）静息状态（即基线测定）。要求测试对象睁眼或者闭目，尽可能放松。在这种静息状态下，脑电中会出现 α 波。此时测定大脑左右半球活动的不对称性，作为静息时的基线记录，以便在后面进行的执行思维任务的脑电中，将这种基线记录的不对称性减去。

2）算术计算任务。要求测试对象心算某个复杂的乘法计算，不能说出来，也不能有其他的肌肉活动。

3）几何任务。给测试对象 30s 的时间，仔细观察一个三维几何物体图形，等图形消失之后，要求测试对象想像这个物体绕某个轴转动的情景。

4）写信任务。要求测试对象在心里默写一封给朋友或家人的信，不能说出来。整个测试过程重复数次，每次重新开始时都要求测试对象从前一次被中断的地方接着往下写。

5）视觉计数任务。要求测试对象看黑板上依次写出的数字，将它们一个个加起来，计算总数。每次写出一个新的数字时，前一个数字被擦除。与其他任务的要求相同，测试对象也不能说，只能看数字。视觉计数任务也重复数次，每次重新开始时要求测试对象在前一次求和的基础上继续加和。

表 10.2 列出了参加这 5 项思维任务测试的 7 位对象的情况。

表 10.2 Keirn（1988）研究工作中测试对象的数据

对象序号	年龄	偏手性	性别	实验次数
1	48	左利手	男	2
2	39	右利手	男	1
3	<30	右利手	男	2
4	<30	右利手	男	2
5	<30	右利手	女	3
6	<30	右利手	男	2
7	<30	右利手	男	1

测试过程中，尽可能避免发出声音并避免身体的其他活动。每次实验中，每个任务重复 5 次，从表 10.2 可知，共进行了 13 次实验，因此，总共有 325 次测试数据，即：

$$13\text{ 次实验} \times 5\text{ 项任务} \times 5\text{ 次测试} = 325\text{ 次测试}$$

整个数据表用 MATLAB 的元胞数组变量存放，第 2 章已经学过，元胞数组是 MATLAB 的一种数据结构，用于组织相互关联但类型不同的数据。本实验的数据包括各次试验的 EEG 记录数据和一些说明注释，由以下 4 个元胞元素组成：

1）测试对象序号；

2）思维任务类型；

3）重复试验序号；

4）EEG 采样数据。

每段 EEG 采样数据长 10s，采样频率为 250Hz，共有 6 通道记录数据，即 C3、C4、O1、O2、P3 和 P4。另外加上由 A1 和 A2 两个电极之间测得的眼动电位 EOG。因此，每次试验共有 17500 个采样数据，即

$$250 \text{ 采样数}/\text{s} \times 10\text{s} \times 7 \text{ 通道} = 17500 \text{ 个采样数据}$$

EEG 采样数据经过频率范围为 0.1 ~ 100Hz 的带通滤波。EEG 数据中包含的频率成分分为：

1）θ 波：2 ~ 7Hz；

2）α 波：7 ~ 13Hz；

3）低频 β 波：14 ~ 20Hz；

4）高频 β 波：大于 20Hz，一般到 64Hz 为止。

这里的问题是通过分析数据，确定在执行这些思维任务时大脑两个半球的活动是否有差别。先推测一下可能在各项思维任务中出现脑电活动差别的区域：

1）静息状态基线测定时应该没有什么差别。

2）执行文字任务时，大脑左半球枕部区域的活动可能会较强，即 O1 与 O2 之间可能会有差别。

3）执行数学计算任务时，大脑左半球顶部区域的活动可能会较强，即 P3 与 P4 之间可能会有差别。

那么，如何定量描述大脑的活动呢？当然，描述的量要尽可能少，但是，应该包含区别各种思维状态所需的信息。下面的例 10.3 利用 MATLAB 程序计算了这批 EEG 脑电数据的频率成分，并用这种频率分析方法测定各种思维状态下的差别。

如果脑电信号主要包含 7 ~ 13Hz 的 α 频率成分，那么测试对象一定是处于休息状态，没有执行什么思维任务；如果测试对象在执行某项思维任务，那么他的脑电信号的主要能量会移到 β 频带。当然，与例 9.9 一样，分析脑电信号频率成分时，需要先消除 60Hz 的噪声干扰，也要消除眨眼引起的肌电干扰。60 Hz 的噪声干扰处于较高的 β 频带，而肌电干扰会出现在 δ 频带和 θ 频带。

例 10.3　左右两个大脑半球活动的差别。

利用上述给定数据，分析 EEG 信号的频率成分，确定各种思维任务引起的不同频带上的两个大脑半球之间 EEG 信号的差别。

解：

MATLAB 程序如下：

```
% constants
```

```
TRIALS = 5;
EXPERIMENTS = 5;
SESSIONS = 13;
SR = 250; % sample rate
% subject/trial map
SUBJ{1} = 1:50;
SUBJ{2} = 51:75;
SUBJ{3} = 76:125;
SUBJ{4} = 126:175;
SUBJ{5} = 176:250;
SUBJ{6} = 251:300;
SUBJ{7} = 301:325;

load eegdata.mat
for i = 1:length(data) data{i}{4} = double(data{i}{4}); end;
datamag = data; % create copy of data

f = SR * (0:2500/2)/2500; % frequency data

for m = 1:length(data)
    kernel = fft(datamag{m}{4}(7,:));
    kernel = kernel/max(kernel);
    datamag{1};
    for (n = 1:6)
        z = abs(fft(data{m}{4}(n,:)).*(1-kernel)); % remove eog
        z = z(1:length(f));
        z = bpf(59.5, 60.5, SR, z, 'f', 1); % apply bandstop filter at
        60Hz (see bpf.m)
        datamag{m}{5}(n,:) = abs(z); % power series for trial
    end
end

clear avgdata;
% average over all experiments
for m = 1:EXPERIMENTS
```

```
    trials = [ ];
for n = 1:SESSIONS
        trials = [trials (m - 1) * TRIALS + (n - 1) * EXPERIMENTS *
TRIALS + (1:TRIALS)];
    end
    avgdata{m} = averagedata(datamag, trials);

    % by subject
    for i = 1:length(SUBJ)
        subjdata{i,m} = averagedata(datamag, intersect(trials,
SUBJ{i}));
    end

    % by age
    for i = 1:3
        if (i == 1)
            z = [SUBJ{3} SUBJ{4} SUBJ{5} SUBJ{6} SUBJ{7}]; % age <
30
    elseif (i == 2)
            z = SUBJ{2}; % age30 - 40
    else
            z = SUBJ{1}; % age40 - 50
    end
    agedata{i,m} = processavg(averagedata(datamag, intersect(tri-
als, z)), SR)
    end

    % by handedness
    for i = 1:2
        if (i == 1)
            z = SUBJ{1}; % LH
        else
            z = setdiff(1:EXPERIMENTS * SESSIONS * TRIALS, SUBJ{1});
% RH, all not in LH
        end
```

```
        handdata{i,m} =processavg(averagedata(datamag, intersect
(trials, z)), SR);
    end

    % by gender
    for i =1:2
        if (i ==1)
            z =SUBJ{5}; % female
        else
            z =setdiff(1:EXPERIMENTS * SESSIONS * TRIALS, SUBJ{5});
% male (setdiff == > everyone else)
          end
          genderdata{i,m} =processavg(averagedata(datamag, inter-
sect(trials, z)), SR);
    end
end
trialdiff =processavg(avgdata, SR);
disp('Done');
end
```

带通滤波等其他几个函数是独立的 MATLAB 脚本，这里没有列出，在本书网站上提供。

注意：以上程序中的多数操作与名为 data 的元胞数组相关，有关元胞数组用法的详细资料希望读者参阅第 2 章以及 MATLAB 的帮助信息。

有关信号滤波处理的基本步骤在例题 9.9 已有说明，这里就不再重复了。EOG 用匹配滤波的方法去除，也就是首先用 FFT 计算 A1 和 A2 之间记录的信号（即 EOG 信号）的频谱，然后从其他 6 路 EEG 记录信号的频谱中减去该频谱成分。

本例题的数据量太大，不能全部显示出来，表 10.3 列出了一种分析结果，其中，差值为正时表示左大脑半球活动强度大，差值为负时则表示右大脑半球活动强度大。任务序号如前所述。

由表中的数据可见，在执行几何任务（任务 3）时，左侧枕部活动强度很大，在执行其他数学计算任务时，左侧枕部也有明显较强的活动。虽然，这些差别应该出现在顶部，但是，由于实验中只用了 6 个记录电极，O1 和 O2 处检测的信号很可能包含了来自顶部区域的成分。

表 10.3　7 位测试对象 5 个任务的大脑两半球之间 EEG 活动之差的分析结果小结（Keirn，1998）
表中的数据是大脑左半球与右半球之间各频带能量百分比的差值

频带	中央（C3，C4）					顶部（P3，P4）					枕部（O1，O2）				
	δ	θ	α	β1	βh	δ	θ	α	β1	βh	δ	θ	α	β1	βh
任务	差值														
1	-1.66	-0.07	1.64	-2.30	-6.12	-0.64	-0.05	1.24	-1.71	-2.70	-0.40	1.56	4.81	1.72	2.84
2	0.72	0.75	-3.86	-3.60	-0.35	0.97	0.39	-5.59	-7.35	1.67	0.70	1.71	2.09	0.76	4.60
3	-1.62	-1.33	-4.12	-3.44	-1.78	-0.62	-0.99	-1.44	-2.19	0.48	0.75	2.68	6.93	4.89	5.73
4	-0.83	1.13	-2.49	-2.64	-0.38	-0.60	0.54	-1.42	-4.87	0.32	1.89	2.09	3.33	1.27	3.74
5	0.06	0.16	-1.84	-2.66	-2.36	-0.14	0.65	-1.32	-3.72	-0.62	0.46	1.95	4.24	2.17	0.93

10.4 糖尿病及其胰岛素治疗

大多数糖尿病患者都需要用胰岛素调节血糖水平。胰岛素或者是口服药片或者是注射剂，它可以通过反馈控制作用来稳定血糖，使血糖浓度保持在正常范围之内。

人体对于血糖的调节能力通常用葡萄糖耐量试验（Glucose Tolerance Test，GTT）来测试。GTT 试验时，病人在空腹的情况下，首先静脉注射一定剂量的葡萄糖，然后，按照一定的时间间隔采集血样，测定血糖含量和胰岛素含量。

要判断这种 GTT 的检测结果是否正常，需要将所测得的数据与标准模型进行比较。葡萄糖调节是一个动态过程，因此，模型具有微分方程组的形式，用于反映血浆葡萄糖浓度随时间的变化。如果实际测得的浓度与模型的预计值相似，那么，医务人员可以认为检测结果正常。

模型从简单的最小系统到复杂的系统具有很多不同的形式，不同模型之间的主要区别就是所采用的房室数目不同。例如：可以只仿真血液的葡萄糖浓度，也可以同时仿真血糖和胰岛素两个浓度；另外，腹腔、肾和胰腺各个部分可以作为独立的系统进行仿真，也可以结合在一起仿真。

Van Riel（2004）提出了一个三房室的最小模型，即：

1）血浆葡萄糖 $G(t)$，单位为 mg/dL；

2）血浆胰岛素 $I(t)$，单位为 μU/mL；

3）组织间隙胰岛素 $X(t)$，单位为 μU/mL㊀。

变量 $X(t)$ 不是生理量，是用于描述胰岛素活动的一个参数，表示腹腔、肾和胰腺这 3 部分组成的组织间隙这个房室中的胰岛素。如下两个微分方程用于描述这个三房室系统：

$$\begin{aligned}\frac{\mathrm{d}G(t)}{\mathrm{d}t} &= k_1(G_b - G(t)) - X(t)G(t)\\ \frac{\mathrm{d}X(t)}{\mathrm{d}t} &= k_2(I(t) - I_b) - k_3X(t)\end{aligned} \tag{10.1}$$

其中，$G(t_0) = G_0$ 为血浆葡萄糖的初始浓度，组织间隙胰岛素的初始浓度 $X(t_0)$ 为 0。第一个方程中的 G_b 表示空腹时血糖的基础水平。如果血糖浓度低于基础水平，则葡萄糖会进入血浆；等到血糖浓度超过基础水平，葡萄糖又会离开血浆。GTT 试验中，注射葡萄糖之前，通常要先测定血糖的基础水平。

第二个方程表示胰岛素进入间隙组织的速率为 k_2，离开间隙组织的速率为

㊀ 原文此处为 $\min^{-1}$。—译者注

k_3。如果血浆胰岛素浓度超过基础水平 I_b，则胰岛素会进入间隙组织；如果血浆胰岛素浓度下降到低于基础水平，则胰岛素就离开间隙组织。

该系统的微分方程很容易用 MATLAB 或者 Simulink 求解，下面的例题说明了利用 Simulink 求解的过程。

例 10.4　利用 Simulink 模型仿真血糖调节的过程。

请利用 Simulink 求解式（10.1）的微分方程组，求血糖随时间变化的曲线。常数 k_1、k_2 和 k_3，以及胰岛素变化曲线 $I(t)$ 取自 Van Riel（2004）的报道。

解：

要用 MATLAB 或者 Simulink 由式（10.1）微分方程组求得 $G(t)$ 的解，首先需要确定其中的 4 个未知数。Van Riel 将式（10.1）的第二个方程重写为

$$\frac{\mathrm{d}X(t)}{\mathrm{d}t} = k_3(S_I(I(t) - I_b) - X(t))$$

式中　S_I——胰岛素的灵敏度，$S_I = k_2/k_3$。

Van Riel 在其编写的 MATLAB 脚本中使用了龙格-库塔求解器 ode45 求解这个微分方程组。当然，通过改变该脚本中的参数值以及胰岛素数据，也可以用这个程序灵活地仿真其他不同的情况。不过，用如下所示的几个 MATLAB 脚本与 Simulink 模型相结合的方法，可以更方便地模拟不同的模型系统。其中，计算正常血糖动态变化过程的 MATLAB 脚本为

```
% Set the workspace variables for the
% Simulink model VanRiel.mdl
% Basal Glucose and Insulin
Gb =92;
Ib =11;
G0 =279;
% Model constants
SI =5e -4;
k3 =0.025;
k1 =2.6e -2;
```

这里的常数以及胰岛素变化曲线取自 Van Riel（2004）的报道。Simulink 模型如图 10.8 所示，注意，常数 G_b 和 I_b 为 MATLAB 变量，积分器 dG/dt 的初始值为 MATLAB 变量 G0。3 个增益模块分别表示常数 k_1、S_I 或 k_3。

现在，只要设置好模型中胰岛素浓度 $I(t)$ 这个输入源模块的参数，就可以运行这个仿真模型。$I(t)$ 输入源是一个二维数组，第一列为时间，第二列为胰岛素数据。先用如下脚本将数据输入到 MATLAB 工作空间的变量 It 中，再通过

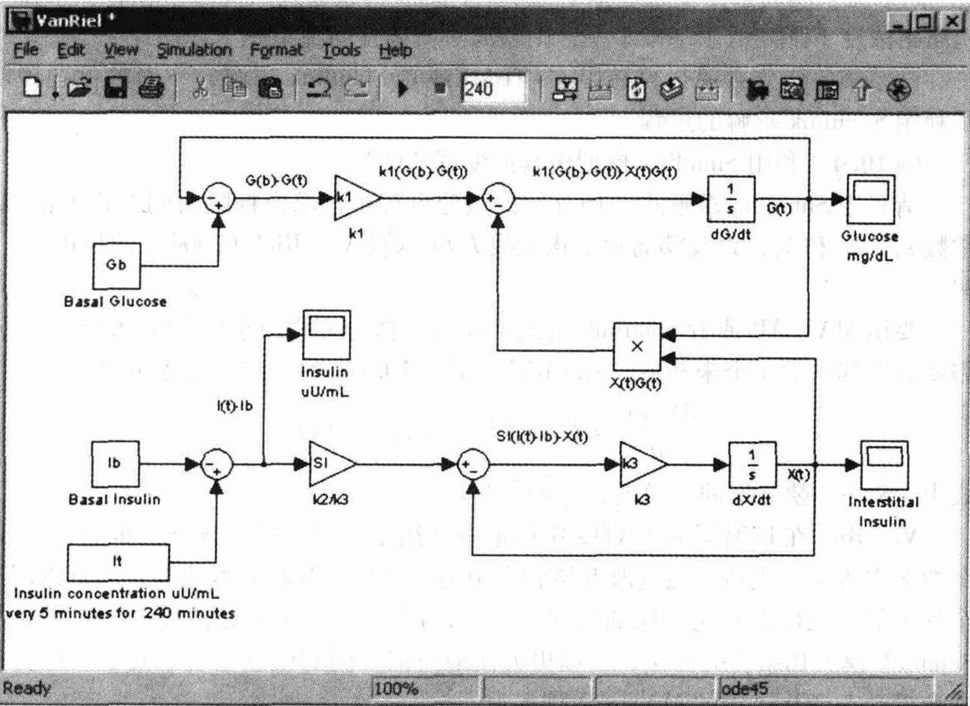


图 10.8 血糖调节的 Simulink 模型

“From Workspace” 模块将 It 数据调入 Simulink 模型。本例题 It 数组设置为

```
%
% Time course of insulin I(t)
%
t_insu=[0 5 10 15 20 25 30 40 60 80 100 120 140 160 180 240];
u=Ib+[0 100 100 100 100 0 0 0 0 0 0 0 0 0 0 0];
It=[t_insu'u'];
```

注意，起始7个数据样本的时间间隔为5min，之后样本采样的时间间隔加长，最后的采样时间为第240min。但是，仿真器的时间步长必须为常数，因此，Simulink 必须插值补缺。另外，如果输入数据的时间比仿真时间短，也可以通过“From Workspace” 模块设定补足数据的方式。如图 10.9 所示为胰岛素浓度“From Workspace” 的参数设置。其中，数据（data）设为 MATLAB 变量 It，采样时间设为5（即5min），并选中插值补缺功能。该窗口的最后一项用于选择补足数据的方式，图中被选中的方式表示：如果设置的仿真时间比所提供的数据长，那么后面输出的值都与最后一个数据元素的值相同。

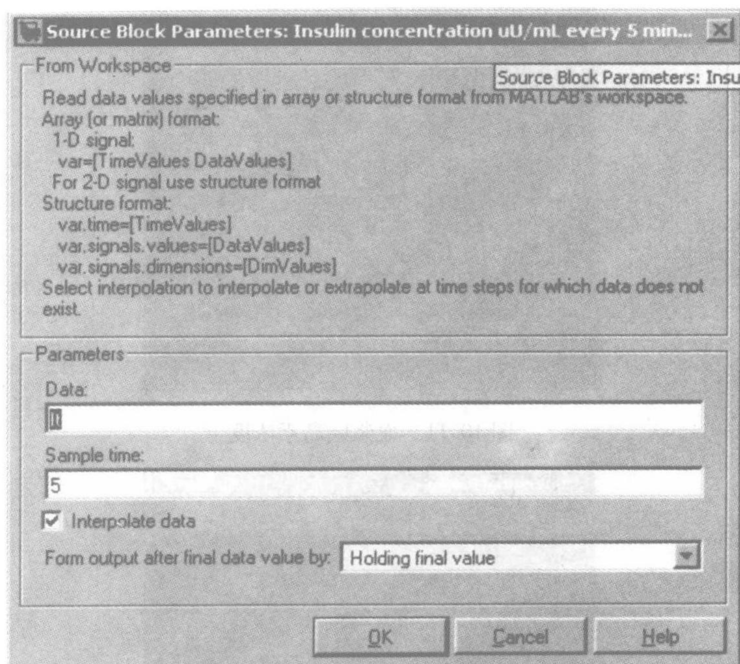


图 10.9　图 10.8 的 Simulink 模型中胰岛素浓度输入源模块的参数设置

最后，将仿真时间设置为从 0.0 开始，到 240.0min 结束。运行仿真之后，模型中 3 个示波器的仿真输出分别如图 10.10 ~ 图 10.12 所示。

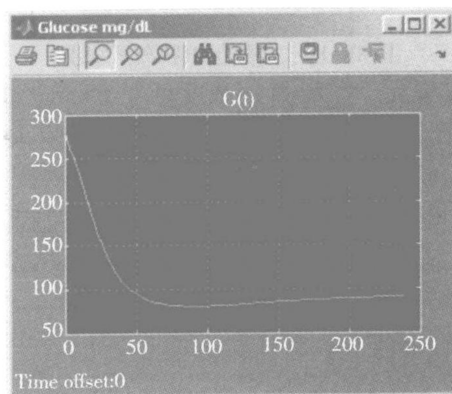


图 10.10　血糖浓度

GTT 试验中测得的样本数据就是要与这样的血糖浓度变化曲线进行对比。注意，大约在注射葡萄糖 1h 之后，紧跟着胰岛素的分泌，血糖浓度降到基础水平以下。在这个正常情况的模拟结果中，最低血糖浓度为 80 mg/dL。

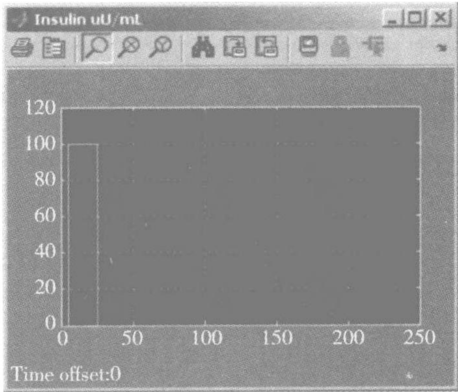


图 10.11　血浆胰岛素浓度

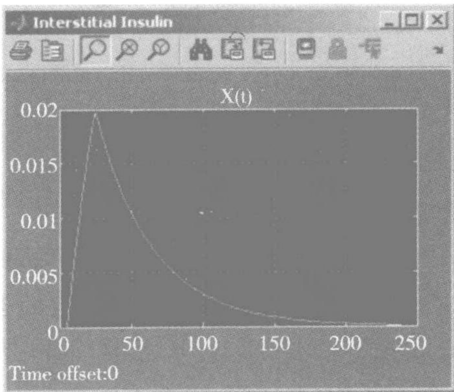


图 10.12　间隙组织胰岛素浓度

对于正常的健康成年人，如果血糖浓度低于 65 mg/dL，其思维能力就会受到影响，这称为低血糖反应。如果血糖浓度正常，但病人表现出精神状态失常，思维能力下降，则称为餐后综合症。

上述模型也可以用于模拟糖尿病患者的血糖调节过程。此时，设置模型参数的 MATLAB 脚本为

```
% Set the workspace variables for the
% Simulink model VanRiel.mdl
% Basal Glucose and Insulin
Gb = 92;
Ib = 11;
G0 = 365;
% Model constants
```

```
SI =0.7e -4;
k3 =0.01;
k1 =1.7e -2;
```

设仿真时间仍为240min，胰岛素输入数据也相同，则仿真输出的血糖浓度、胰岛素浓度和间隙组织胰岛素分别如图 10.13～图 10.15 所示。

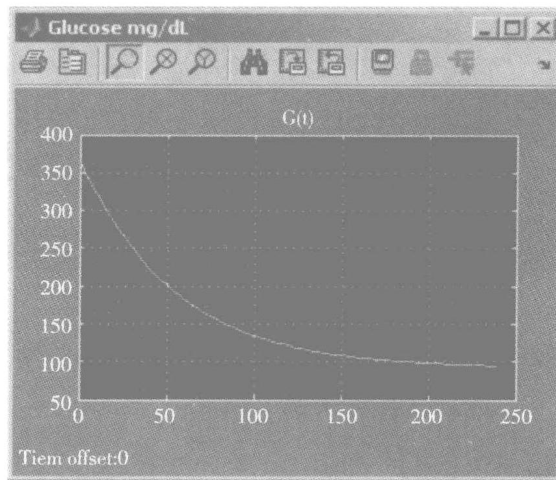


图 10.13 血糖浓度

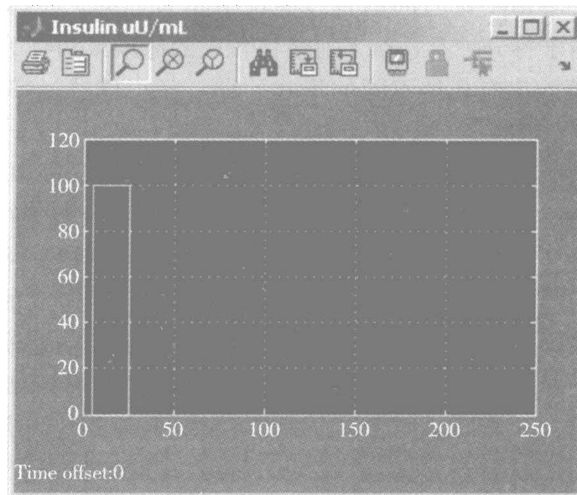


图 10.14 血浆胰岛素浓度

美国国立卫生研究院（NIH-NIDDK）诊断糖尿病的标准是，在摄入葡萄糖 2h 之后，血糖浓度大于或等于 200 mg/dL。以上仿真得到的血糖浓度曲线虽然超

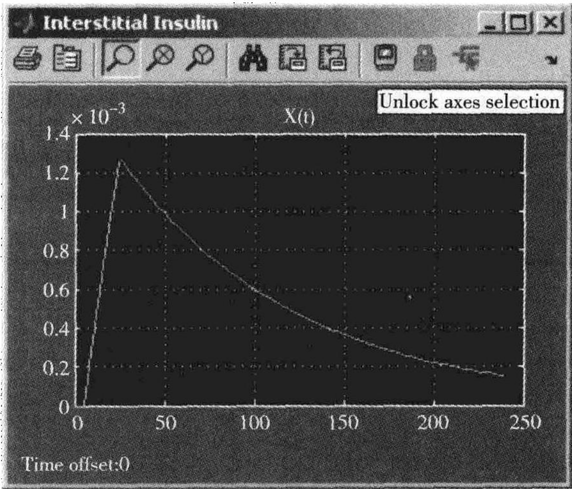


图 10.15　间隙组织胰岛素浓度

出了正常水平，但还未达到糖尿病诊断标准，因此，可以诊断为糖尿病前期，或者是糖耐量异常。

10.5　肾清除率

肾清除率 Cl_R 是指单位时间内肾可以排出相当于多少体积血浆中的某种物质，也就是，相当于每分钟有这么多血浆体积中的尿素或药物等某种给定物质被“完全清除”。肾清除率有几种不同的形式，即：肾小球滤过率、有效肾血浆流量和肾小管吸收率。

评价肾动态功能的方法有很多种，例如，有些是测量尿和血浆（或者只是血浆）中某种放射性核素（即同位素）随时间的浓度变化；有些是在血浆同位素排泄过程中用伽马射线摄像机测量两个肾中的同位素浓度。将临床检测所得到的数据与正常模型的动态仿真结果进行比较，就可以评价肾功能的状况。这里所述的临床检测方法用中心和外周两个房室就可以进行简单的建模仿真。

Estelberger 和 Popper 两人在 2002 年建立了这种两房室模型，该模型具有两个功能独立的房室，即充分灌注的中心房室和部分灌注的外周房室。图 10.16 所示是该模型的框图。放射性标记物的传输过程可以间接地从两个房室的同位素浓度变化中测得，整个传输过程由同位素注射、两房室之间的传输以及肾清除过程组成。

该传输模型可以用如下两个联立微分方程表示，它们分别描述了两个房室中的同位素浓度变化，即

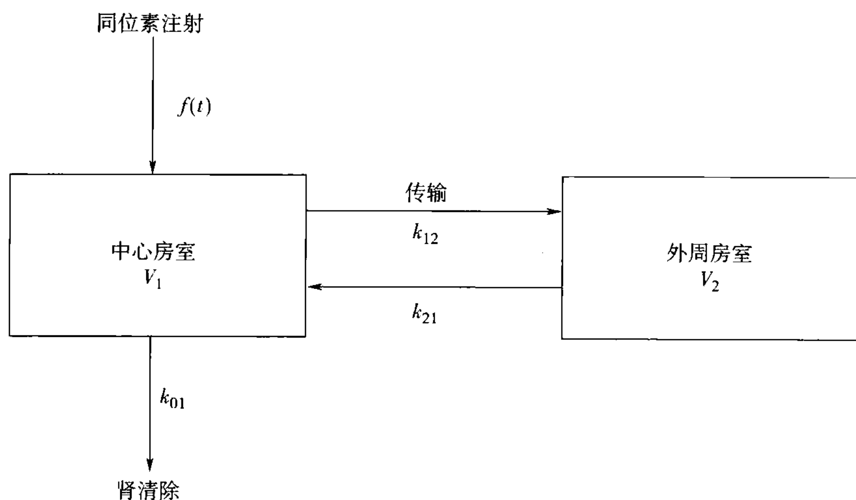


图 10.16　模拟肾清除过程的一种两房室模型（摘自 Estelberger 和 Popper，2002）

$$\frac{\mathrm{d}c}{\mathrm{d}t} = d(t) - (k_{01} + k_{21})c + k_{12}p$$

$$\frac{\mathrm{d}p}{\mathrm{d}t} = k_{21}c - k_{12}p \tag{10.2}$$

其中

$$d(t) = \frac{D}{T}, 0 \leqslant t < T \tag{10.3}$$

表示同位素团注在 T 秒时间内完成。作为例题，这里假设同位素的注射是匀速的，并假设中心房室 V_1 的体积已知。

Estelberger 和 Popper 所写的论文还说明了如何从实验数据中估计模型群体参数，其优化处理过程超出了本书的范围，并且与下面的例题关系不大，有兴趣的读者可以查阅原文。

与上一节采用的仿真方法不同，下面不用 Simulink 而用 MATLAB 程序来求解这个微分方程组。

例 10.5　肾清除率的仿真计算

请求解式（10.2）系统的两个微分方程，其中，初始条件由式（10.3）给定，常数为 $k_{01} = 0.0041, k_{12} = 0.0585, k_{21} = 0.0498, V_1 = 7.3, c(0) = p(0) = 0$。

解：

本题将说明 MATLAB 全局变量的用法。式（10.2）微分方程反映的是质量守恒定理，设同位素注射在 0 时刻开始，每次试验的注射时间长短不等。

模型的参数和同位素团注数据被设定为全局变量，下面的 MATLAB 作图程

序以及求解微分方程组的程序都要使用这些全局变量。

首先，按照 MATLAB 惯例，将式（10.2）的系统微分方程组编写为 MATLAB 函数，设时间变量为 t，函数值矢量为 y。其他模型参数都设为全局变量。该函数的文件名为 renal. m，程序如下：

```
function yprime = renal(t,y)
% separate the components of the state:
% central compartment concentration and
% peripheral compartment concentration
c = y(1);
p = y(2);
%
global D tau;
global k01 k12 k21 V1;
% let time t be in minutes
% now compute f(t)
%
if t < = tau
    f = D/tau;
else f = 0;
end
%
dcdt = (f - (k01 + k21) * c + k12 * p)/V1;
dpdt = k21 * c - k12 * p;
yprime = [dcdt,dpdt]';
```

下面的 MATLAB 脚本程序用 3 次不同的同位素注射数据，分别求解系统微分方程组，并用不同形式的线条将所求得的中心房室的 3 条同位素浓度变化曲线显示在同一张图上。作图方法是：作出第一条曲线后，在程序中加一条 hold on 指令，就可以将另外 2 条曲线也画在同一张图上。脚本程序如下：

```
% Example 10.5 Renal Clearance
global D tau;
global k01 k12 k21 V1;
% parameters
k01 = 0.0041;
```

```
k12 =0.0585;
k21 =0.0498;
V1 =7.3;
% initial value conditions
c0 =0;
p0 =0;
%
D =2500;
tau =0.5;
% use ode45
[t,y] =ode45(@ renal ,[0:0.5:240],[c0 p0]);
plot (t,y(:,1),'b.');
xlabel ('Time t (min)')
ylabel('Isotope concentration c(t)')
hold on
%
D =2500;
tau =10.0;
% use ode45
[t,y] =ode45(@ renal,[0:0.5:240],[c0 p0]);
plot(t,y(:,1),'g:');
%
D =2500;
tau =240;
% use ode45
[t,y] =ode45(@ renal,[0:0.5:240],[c0 p0]);
plot(t,y(:,1),'r--');
hold off
```

以上程序中仿真时间总长均为240min，时间步长为0.5min。如图10.17所示为3次同位素注射试验的求解结果。注意，前2个解的同位素浓度变化轨迹很接近，它们仿真的是系统对于同位素团注脉冲输入的响应；而第3个解（虚线轨迹）仿真的是系统对于斜坡输入（0～240min）的响应，这种响应在240min仿真时间内不表现反馈控制作用。

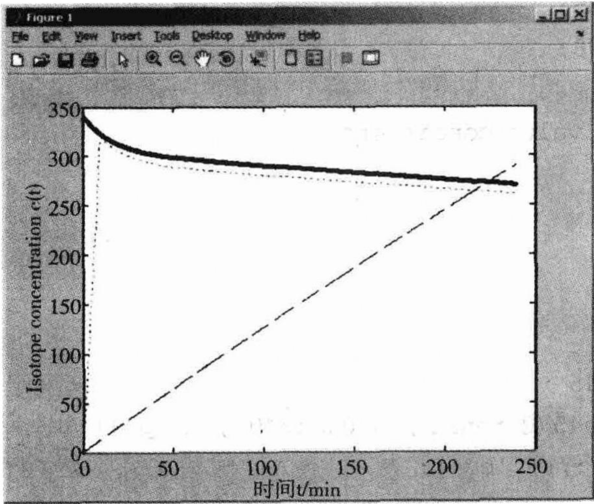


图 10.17 肾清除率程序的求解结果（实线和点线分别为 0.5min 和 10min 团注脉冲输入的响应，虚线为注射过程持续了 240min 时的响应）

10.6 配准问题以及运动分析

要从一组图像序列中估计物体的运动过程，其关键是要根据运动目标上几个特征标志点的对应关系，即配准，分析出前后两张图像中物体相对运动的关系。特征点必须是惟一识别的，并且，通常假设物体是刚性的。

人体运动的步态分析使用髋部和腿部特征点位置的记录数据。例如，在这些特征部位安装发光二极管（LED）标记物，然后让测试对象在暗室里行走，记录其走动时的状态图像用于分析。许多康复机构都配备有步态测试实验室，可以进行此类运动记录和分析，并将结果用于医疗和诊断。

图 10.18 显示了利用髋部和腿部的特征点配准来估计旋转矩阵的方法，下面例题的数据由 Purdue university 的 Charles Krousgrill 教授慷慨提供。

例 10.6 刚性人体特征点的运动分析

如图 10.18 所示，在髋部、大腿和小腿分别粘贴 6 个标记物，这些标记点的局部坐标列于表 10.4。

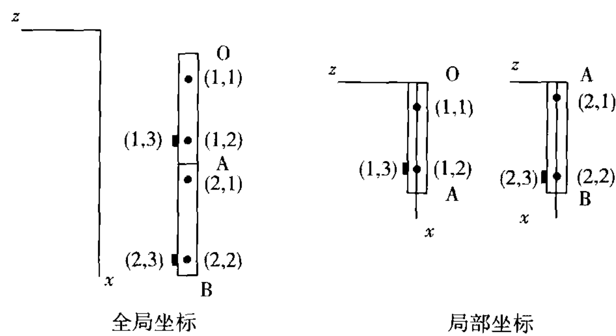


图 10.18　人体髋部和腿部步态分析标记点的坐标系统（Krousgrill 提供，2005）㊀

表 10.4　人体髋部和腿部步态分析标记点的局部坐标（由 Krousgrill 提供，2005）

标　记　点	x 坐标	y 坐标	z 坐标
(1，1)	6	4	0
(1，2)	12	4	0
(1，3)	12	0	4
(2，1)	6	3	0
(2，2)	12	3	0
(2，3)	12	0	4

从某一帧运动图像中采集的这些标记点的坐标以及髋关节 O［Hip（O）］（即股骨头）的坐标数据列于表 10.5，这些是三维全局坐标（global coordinate）。

表 10.5　刚性人体移动之后髋部和腿部步态分析标记点的全局坐标（由 Krousgrill 提供，2005）

标　记　点	x 坐标	y 坐标	z 坐标
Hip（O）	2	1	0
(1，1)	3.18	7.13	3.60
(1，2)	7.16	10.47	6.60
(1，3)	8.89	5.88	9.41
(2，1)	9.11	15.45	10.44
(2，2)	6.99	20.66	12.52
(1，1)	9.36	19.88	16.86

假设大腿和小腿的长度均为 18.13in，请用以上数据计算出大腿和小腿的旋转矩阵，并计算踝关节 B 点的三维全局坐标。

㊀　图中，O、A 和 B 三点分别对应于髋关节、膝关节和踝关节。—译者注

髋部和腿部标记点的位置是用局部坐标定义的，也就是相对于某个标记点的三维位置。此例中，髋关节 O 是大腿的参考标记点，膝关节 A 则是小腿的参考标记点（见图 10.18）。局部坐标数据见表 10.4。

但是，运动过程不能用局部坐标系统记录，只能用步态实验室的全局坐标系统记录。摄像机拍摄下人体的走动图像，利用图像中标记点的位置可以确定人体相对于实验室摄像机的运动位置，而不是相对于髋部或腿部参考点的运动位置。因此，要计算大腿和小腿的转动，首先要确定全局坐标和局部坐标之间的转换关系。

求解这个问题的一般步骤是：将大腿和小腿的全局坐标记录数据转换为局部坐标；通过标记点的配准来确定相对于初始静止状态的转动。并且，必须先计算大腿的转动，因为计算小腿转动时，需要用到大腿转动的数据。

运动分析时，需要跟踪 3 组坐标数据：移动之前，静息状态时大腿和小腿的局部坐标；移动之后，每个标志点的全局坐标；计算得到的移动之后的大腿和小腿的局部坐标。

首先，利用表 10.4 和表 10.5 的数据初始化 MATLAB 变量，即

```
% Example 10.6
% At rest (in local coordinates)
% Each column vector is the 3D coordinates
% of a marker (initial conditions)
ThighRestLocal =[6 4 0;12 4 0;12 0 4]';
LegRestLocal =[6 3 0;12 3 0;12 0 4]';
ThighFinalGlobal = [3.18 7.13 3.60 ;7.16 10.47 6.6;8.89 5.88
9.41]';
LegFinalGlobal =[9.11 15.45 10.44;6.99 20.66 12.52;9.36 19.88
16.86]';
% Column vectors of positions of O,A and B
HipPos =[2 1 0]';
ThighLen =[18.13 0 0]';
LegLen =[18.13 0 0]';
```

移动之后大腿位置的全局坐标可以表示为如下等式：

$$\boldsymbol{X}_{\text{thigh, motion}} = \boldsymbol{R}_{\text{thigh}} \boldsymbol{X}_{\text{thigh, initial}} + \boldsymbol{O} \tag{10.4}$$

式中 $\boldsymbol{X}_{\text{thigh, initial}}$，$\boldsymbol{X}_{\text{thigh, motion}}$——矩阵，其列矢量为标记点的位置；

$\boldsymbol{R}_{\text{thigh}}$——绕髋关节的旋转；

$\boldsymbol{O}$——移动之后髋关节的三维位置。

于是，$\boldsymbol{R}_{\text{thigh}}$可以计算如下：

$$\boldsymbol{R}_{\text{thigh}} = (\boldsymbol{X}_{\text{thigh,motion}} - \boldsymbol{O})(\boldsymbol{X}_{\text{thigh,initial}})^{-1} \tag{10.5}$$

其 MATLAB 计算程序为

```
% Find rotation of thigh about hip
% First,determine coordinates of thigh marders
% with respect to the hip
ThighFinalLocal = ThighFinalGlobal - [HipPos HipPos HipPos];
% Rotation computed by correspondence matching
% and is in the local coordinate system centered at the Hip
RotThigh = ThighFinalLocal * inv(ThighRestLocal);
```

计算小腿的旋转要难一些，小腿是绕膝关节转动的，而膝关节又绕髋关节转动，因此，小腿的运动由两个转动组成，即

$$\boldsymbol{X}_{\text{leg,motion}} = \boldsymbol{R}_{\text{thigh}}\boldsymbol{R}_{\text{leg}}\boldsymbol{X}_{\text{leg,initial}} + \boldsymbol{A} \tag{10.6}$$

其中

$$\boldsymbol{A} = \boldsymbol{O} + \boldsymbol{R}_{\text{thigh}}[18.13\ 0\ 0]^{\mathrm{T}} \tag{10.7}$$

将式（10.6）变换为

$$\boldsymbol{R}_{\text{leg}} = \boldsymbol{R}_{\text{thigh}}^{-1}(\boldsymbol{X}_{\text{leg,motion}} - \boldsymbol{A})(\boldsymbol{X}_{\text{leg,initial}})^{-1} \tag{10.8}$$

其 MATLAB 的计算程序为

```
% Find final position of knee
KneePos = ThighLen;
FinalKneePos = (RotThigh * KneePos) + HipPos;
% Find rotation of leg about knee
% First determine local coordinates of leg markers
% with respect to the knee
% The rotation is in the local coordinate system of the knee
LegFinalLocal = LegFinalGlobal - [FinalKneePos FinalKneePos
FinalKneePos];
CombinedRotLeg = LegFinalLocal * inv(LegRestLocal);
RotLeg = inv(RotThigh) * CombinedRotLeg;
```

最后，计算踝关节的位置。踝关节处没有标记物，移动之后其全局坐标位置就是小腿末端（见图 10.18 的 B 点）的转动位置，利用踝关节绕膝关节的转动，计算为

$$\boldsymbol{B} = \boldsymbol{R}_{\text{thigh}}\boldsymbol{R}_{\text{leg}}[18.13\ 0\ 0]^{\text{T}} + \boldsymbol{A} \tag{10.9}$$

MATLAB 程序为

```
% Coordinates of the ankle are
% Ankle = Knee + Rot * Ankle at Rest
% Ankle is in the local coordinates of the
% Knee ,which is the center of rotation
AnklePos = LegLen;
FinalAnklePos = FinalKneePos + CombinedRotLeg * AnklePos;
```

由这些程序计算得到的大腿和小腿的旋转矩阵分别为

```
>> RotThigh
RotThigh =
    0.6633    -0.7000    -0.2675
    0.5567     0.6975    -0.4500
    0.5000     0.1500     0.8525

>> RotLeg
RotLeg =
    0.4233    -0.8933     0.1581
    0.9063     0.4183    -0.0749
   -0.0011     0.1747     0.9864
```

踝关节的位置为

```
>> FinalAnklePos
FinalAnklePos =
     7.6203
    26.8353
    15.3501
```

由于旋转变换不改变大腿和小腿的长度，因此旋转矩阵应该是正交的。如果 $\boldsymbol{R}^{\text{T}}\boldsymbol{R}$ 是一个单位矩阵 $\boldsymbol{I}$，那么矩阵 $\boldsymbol{R}$ 就是正交的。下面来验证大腿和小腿的旋转矩阵是否正交，计算如下：

```
>> RotThigh' * RotThigh
```

```
ans =
    0.9999    -0.0011    -0.0017
   -0.0011     0.9990     0.0013
   -0.0017     0.0013     1.0008

>> RotLeg' * RotLeg
ans =
    1.0007     0.0007    -0.0020
    0.0007     1.0035    -0.0003
   -0.0020    -0.0003     1.0036
```

可见，在数值计算的误差范围内，以上两个矩阵的计算结果可以认为是单位矩阵 $\boldsymbol{I}$。请参考第 4 章的内容，分析矩阵求逆时误差是如何传播的。

10.7　PHYSBE 仿真系统

PHYSBE 是用于模拟血流中的氧、营养物质、热量和化学示踪剂传输过程的一个人体循环系统模型（McCleod，1966，1968）。虽然 PHYSBE 的基础工作源于 20 世纪 60 年代，但自从在 Simulink 中实现了其仿真之后，才可以方便地用于教学和科研。在 MathWorks MATLAB 主网站上可以下载 PHYSBE 的 Simulink 实现，本书附录 B. 4. 3 介绍了有关 Simulink 版 PHYSBE 模型的安装以及使用的方法。本节将举例说明如何应用 PHYSBE 模拟心血管系统疾病并预测这些疾病对于血流的影响。

例 10.7　正常循环系统的 PHYSBE 仿真

请按照附录 B. 4. 3 的说明，安装并运行 PHYSBE 的仿真。

解：

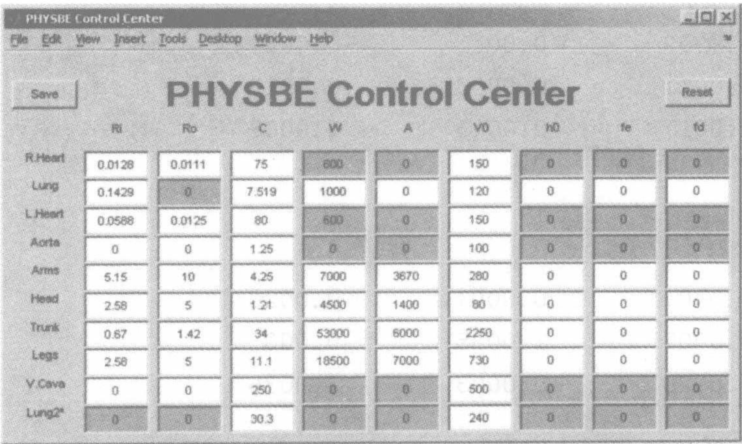
运行 MATLAB 脚本程序 pctrl2，可以启动 PHYSBE 仿真软件，并打开如图 10. 19所示的 PHYSBE 控制面板。注意：仿真时不要改变任何数据的默认值，但要保存好参数文件。

然后，打开如图 10. 20 所示的 PHYSBE 的 Simulink 模型并启动仿真。

仿真结果显示在 Simulink 的示波器中，如图 10. 21 和图 10. 22 所示分别为心脏的血压和血容量。

两个示波器显示的颜色不同，很容易辨别。还有一个选取式示波器，可以用于显示任何其他感兴趣的系统参数。

这些正常循环系统的血压和血容量等模拟结果将在下面几种系统的模拟中作为对照数据。



	Ri	Ro	C	W	A	V0	h0	fe	fd
R.Heart	0.0128	0.0111	75	800	0	150	0	0	0
Lung	0.1429	0	7.519	1000	0	120	0	0	0
L.Heart	0.0588	0.0125	80	600	0	150	0	0	0
Aorta	0	0	1.25	0	0	100	0	0	0
Arms	5.15	10	4.25	7000	3670	280	0	0	0
Head	2.58	5	1.21	4500	1400	80	0	0	0
Trunk	0.67	1.42	34	53000	6000	2250	0	0	0
Legs	2.58	5	11.1	18500	7000	730	0	0	0
V.Cava	0	0	250	0	0	500	0	0	0
Lung2*	0	0	30.3	0	0	240	0	0	0

图 10.19　PHYSBE 控制面板

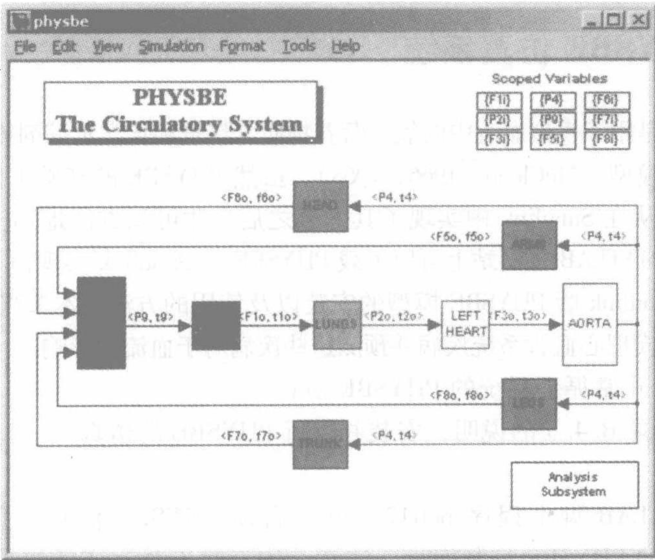


图 10.20　PHYSBE 的 Simulink 模型

10.7.1　主动脉缩窄

主动脉流出管道狭窄引起的病症称为主动脉缩窄，缩窄部位产生压力差，使得该部位前面的血压高于正常值，后面的血压则低于正常值。如图 10.23 所示，主动脉缩窄往往导致人体上身和上肢的血压升高，而下身和下肢的血压降低（Suk，2001）。

如果不进行治疗，主动脉缩窄会引起很多严重的后果，例如导致心肌梗塞、

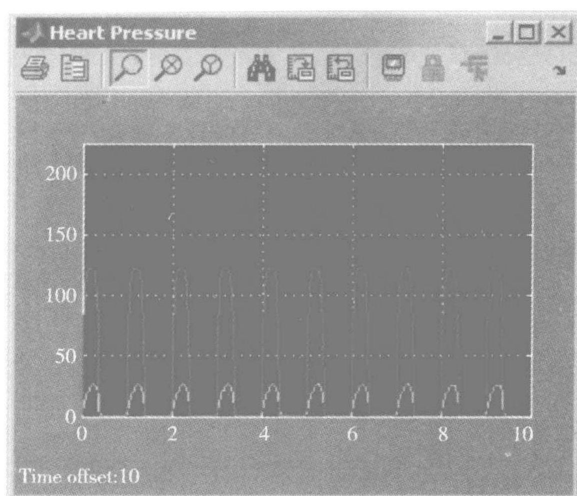


图 10.21　PHYSBE 的 Simulink 模型输出：心脏血压

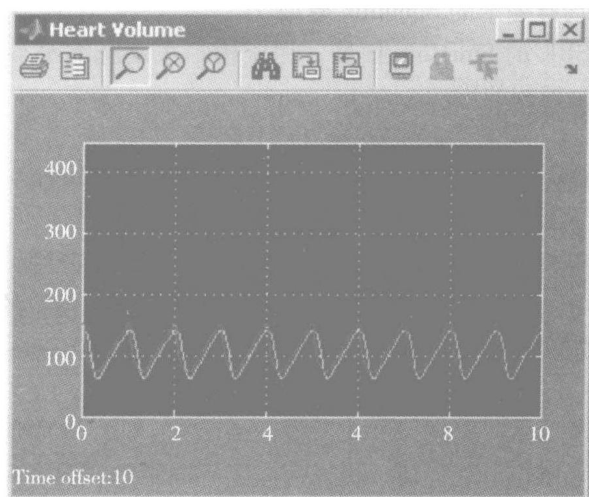


图 10.22　PHYSBE 的 Simulink 模型输出：心脏血容量

颅内出血、心内膜炎感染、冠状动脉疾病以及主动脉破裂等并发症，其死亡率很高。因此，如果任其发展，50 岁之前的死亡率非常惊人，有 90%。幸运的是现代化的医疗技术几乎可以完全消除这种疾病引起的可怕后果。

主动脉缩窄就是主动脉某处的管径被明显“掐”小（Sokolow 和 McIlroy，1981）。由于缩窄部位的血流量与该部位之前和之后流过的血流量必须相等，因此截面积 $A=\pi r^2$ 的减小，必然使血流速度增加，于是，就会形成一个压力梯度，使得躯干下部的压力减小。在 Simulink 仿真模型中，这种减压效果用流通路径上

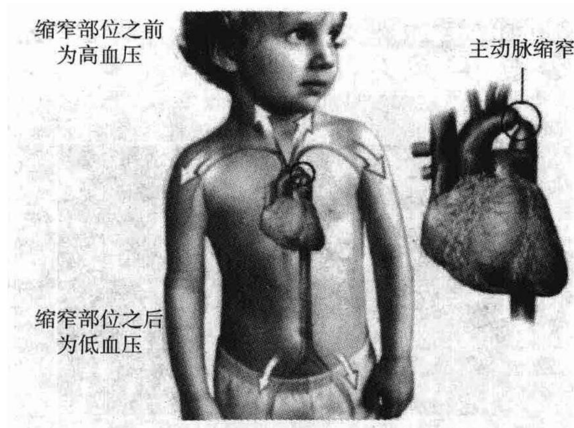


图 10.23 主动脉缩窄（摘自 Suk，2001）

增加一个阻抗来模拟，根据欧姆定律 $Q = \Delta P/R$(Li,2004)，就有了所需要的压力差。

例 10.8 主动脉缩窄的 Simulink 模型

请修改 PHYSBE 的 Simulink 模型，用于模拟主动脉缩窄，并预测这种疾病对于心脏血压和血容积的影响。

解：

PHYSBE 模型只需作两个修改，就是在降主动脉部位添加一个阻抗和一个脉冲延时。

如图 10.24 所示，主要的改动放在一个 Simulink 子系统中，这样既简单又方便。在该子系统中，用一个脉冲延时模块实现脉冲延时，模块的时间延迟参数设为 0.08s，根据文献报道，该参数的变化范围应该为 0.08 ~ 0.1s（Sokolow，1981）。

阻抗用一个增益模块实现，其增益值设为 $1/R$，加在脉冲延时之后，与通向腿部和躯干的动脉血管串联连接。根据病人的不同生理状况、身体条件以及主动脉缩窄的程度，阻抗值可以不同。

如图 10.25 所示，在整个 PHYSBE 模型中，图 10.24 的主动脉缩窄子系统加在升主动脉模块与腿模块之间。

另外，从图 10.24 的主动脉缩窄子系统中可见，狭窄部位前后两端的压力变化值分别保存在新设定的 MATLAB 对象 PA 和 PB 中，用于作图和量化计算。这两个对象由 PHYSBE 的控制分析调用，在修改之后的 PHYSBE 控制中心列表中分别排在第 11 和第 12 列。

图 10.26、图 10.27 和图 10.28 所示为仿真结果，图 10.26 是左、右心室血压，图 10.27 是左、右心室血容积，图 10.28 是主动脉缩窄部位前后两端的血压。

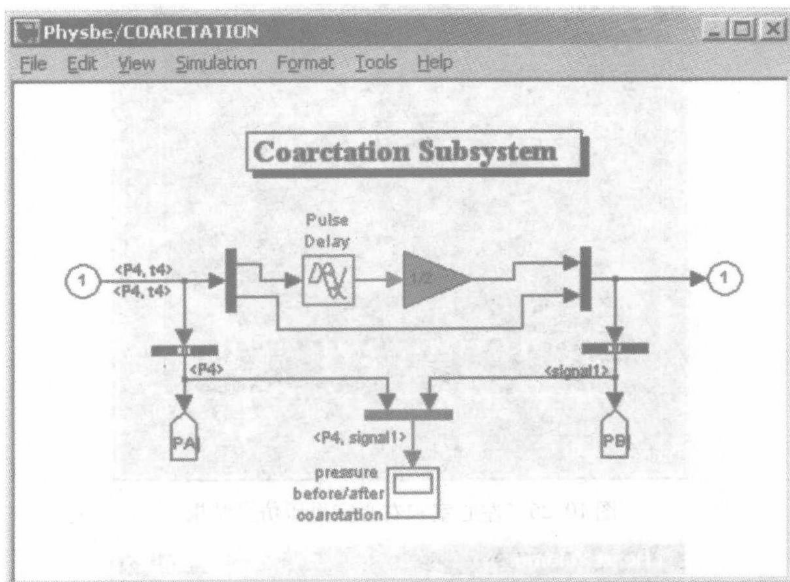


图 10.24　主动脉缩窄子系统

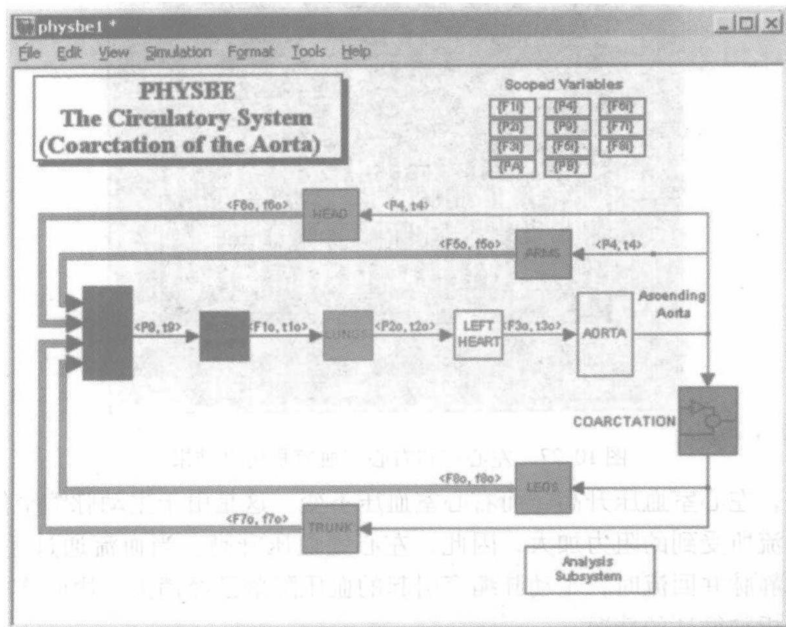


图 10.25　主动脉缩窄子系统（Coarctation）在 PHYSBE 循环系统中所处的位置

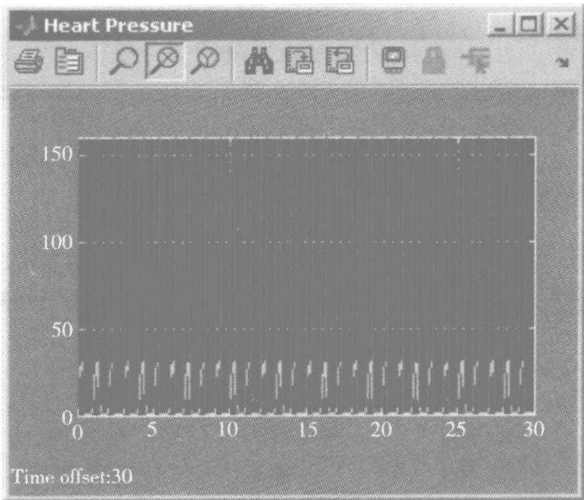


图 10.26 左心室和右心室血压仿真结果

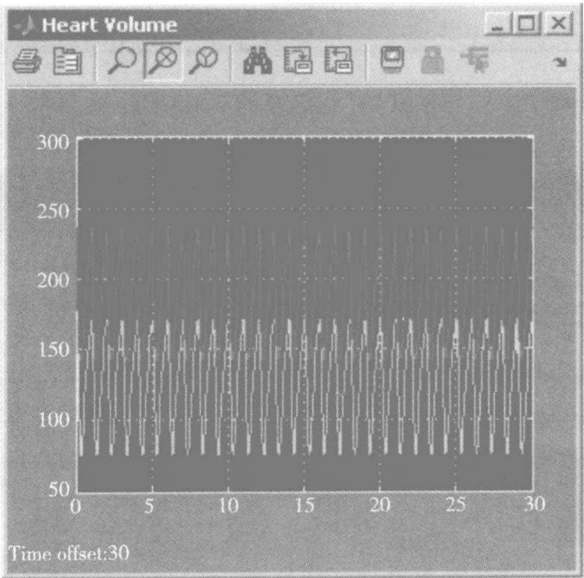


图 10.27 左心室和右心室血容积仿真结果

注意，左心室血压升高，而右心室血压不变。这是由于主动脉缩窄使左心室流出的血流所受到的阻力加大，因此，左心室血压升高。当血流通过毛细血管，再汇集到静脉并回流时，主动脉缩窄引起的血压降落已经消失。用同样的方法可以跟踪分析肺循环的情况。

这些仿真结果说明，动脉血流阻抗的一些小变动会影响全身的血压。不过，PHYSBE 系统的仿真有一定的限制，例如，PHYSBE 模型将两条手臂合在一起，

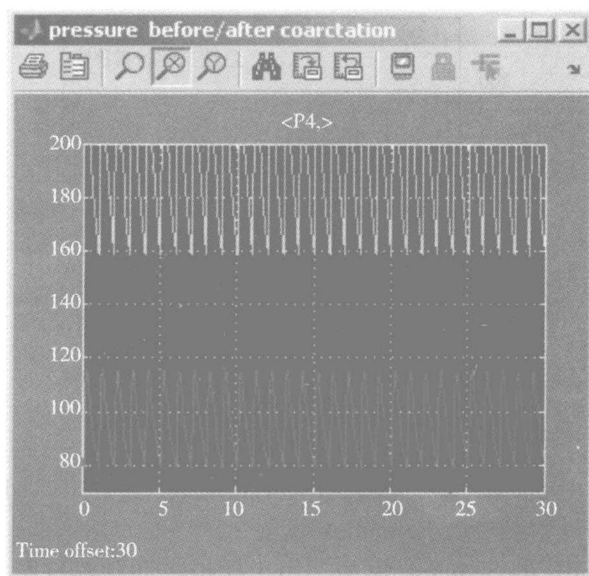


图 10.28　主动脉缩窄部位前后两端的血压

用一个子系统表示，两条腿也是如此，因此，不能仿真主动脉缩窄引起的左、右手臂之间存在的 15mmHg 的血压差（Braunwald，1988）。

10.7.2　主动脉瓣狭窄

主动脉瓣狭窄是主动脉瓣处的沉积物形成阻碍引起的，沉积物导致瓣口狭窄，并使主动脉瓣不能正常开闭，因而流出左心室的血流减少。为了补偿心脏排血量，确保身体所需的血液供应，这会迫使左心室增加工作强度（Texas heart institute，2004）。

主动脉瓣狭窄的病因很多，可由先天性狭窄、风湿热、主动脉瓣钙化等因素造成。少数人出生时其主动脉瓣只有两个瓣叶，而不是正常的三个瓣叶，经过成年累月的磨损，瓣膜出现钙化和损坏，导致其活动性减弱。风湿热也会损伤主动脉瓣的瓣膜，引起瓣膜交界处粘连。另外，随着年龄的增长，人体内主动脉瓣等部位的胶原会受损，形成钙质沉积。主动脉瓣上的钙质沉积物会使瓣膜的活动性降低，从而增加了血流阻力。主动脉瓣狭窄的症状表现为：晕厥、呼吸短促、心率失常、心绞痛和咳嗽等（Texas heart institute，2004）。

主动脉瓣阻力增加会导致左心室血压上升，动脉血压下降。严重的主动脉瓣狭窄会导致心脏的每搏排出量减少、后负荷增加、以及心室收缩末期容积增加。如果主动脉瓣狭窄的仿真模型正确，则其仿真结果应该具有与图 10.29 相似的血压变化曲线。

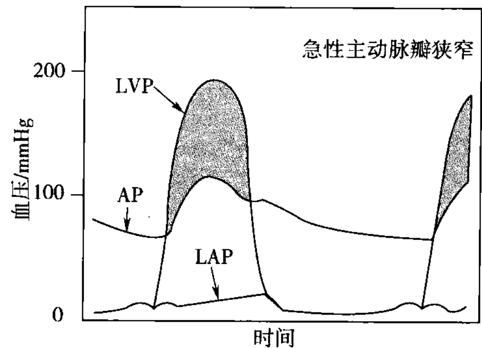


图 10.29 主动脉瓣狭窄时血压随时间变化的曲线（摘自 Klabunde，2004）

例 10.9 主动脉瓣狭窄的 Simulink 模型

请修改 PHYSBE 的 Simulink 模型，用于模拟主动脉瓣狭窄，并预测其对于心脏血压和血容积的影响。

解：

如果直接修改 PHYSBE 模型中的主动脉瓣半径等生理参数，那么仿真就是最准确的。但是，PHYSBE 模型的生理参数是不允许改动的。因此，如果要利用 PHYSBE 仿真主动脉瓣狭窄，可以通过增加左心室阻力 R_0 的方法来实现。出现主动脉瓣狭窄时，瓣膜半径减小，根据泊肃叶定理（Poiseuille law）（Germann，2005），血流阻力与瓣膜半径成反比关系，因此，主动脉瓣狭窄会使血流阻力增加。而血流阻力增加导致的生理效应是血流减少，其数学关系为 $F = \Delta P/R$，其中，F 为血流量，ΔP 为血压差，R 为阻力。下面就用这种血流减少来衡量模型的主动脉瓣狭窄效应。

将左心室阻力 R_0 从 0.0125 mmHg/mL/s 增加到主动脉瓣狭窄所对应的生理值 0.135mmHg/mL/s，该数值由严重主动脉瓣狭窄的数据 180dyne · s 换算得到，请参见 Mascherbauer（2004）的论文。

主动脉瓣狭窄发生时，左心室最大血压超过了 160mmHg（见图 10.30）；心室收缩末期容积，也就是射血后心室中剩余的血容量，增加到 85mL 左右（见图 10.31）；左心室血流输出最大速率降低为 460mL/min（见图 10.33），只有正常值 925mL/min（见图 10.32）的一半。

以上的仿真结果与病理生理过程相同，由于管道半径与阻力成反比关系，主动脉瓣的阻碍引起血流阻力增加。而血流速率与阻力的关系为 $F = \Delta P/R$，阻力 R 的增加使流出心脏的血流速率减小。如果心脏射出的血量不足，血流速率也不够快，那么就不能通过心血管系统把人体所需的氧气和其他营养物质有效地传输

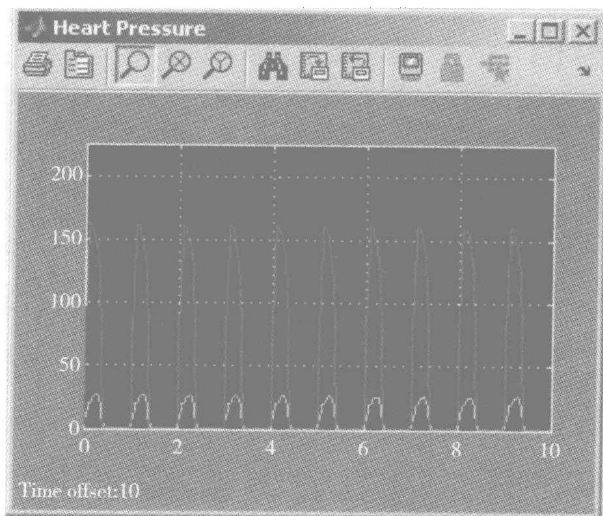


图10.30　主动脉瓣狭窄的血压仿真结果

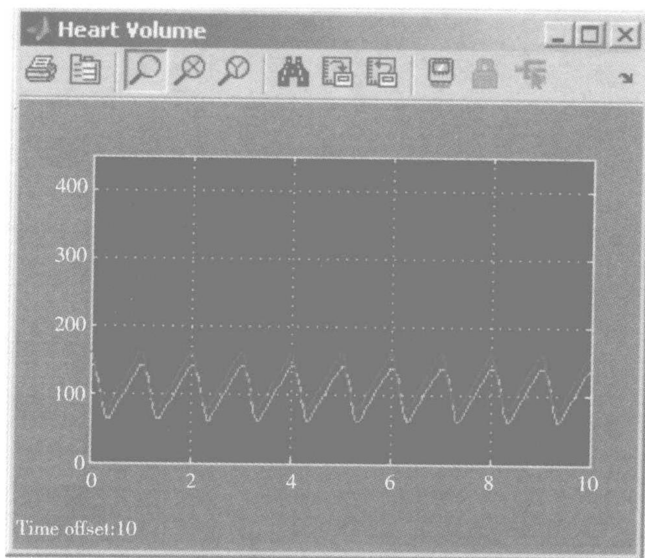


图10.31　主动脉瓣狭窄的心室容积仿真结果

到各个组织中。血流量的减少还会触发人体体内的调节机制，以补偿血流的不足，也就是使左心室泵血强度增加，这样，心脏的劳损速度就要比正常心脏快。

另外，心输出阻力的增加使得血液不能像正常情况那样快速从左心室流出心脏，使心室收缩末期容积增加，也就是心脏收缩结束时滞留在心脏中的血液量增加。

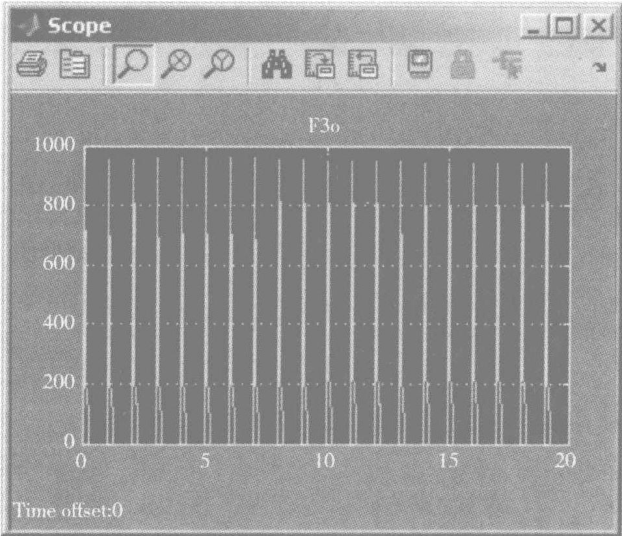


图 10.32　正常心脏左心室血流输出速率的仿真结果

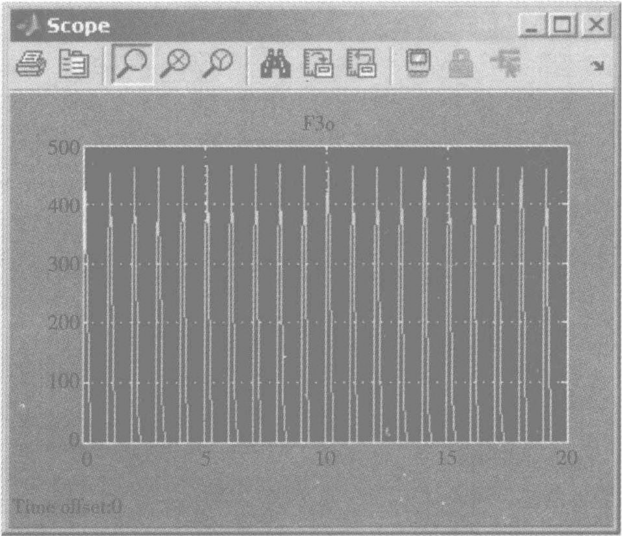


图 10.33　主动脉瓣狭窄的左心室血流输出速率的仿真结果

10.7.3　室间隔缺损

室间隔缺损（Ventricular Septal Defect，VSD）是一种心脏畸形，通常是先天性的，或者与心肌梗塞等其他疾病并存。VSD 的特征是左右两个心室之间的隔膜上存在小孔（Lue 和 Takao，1986），它是最常见的先天性心脏病。VSD 病情

的严重程度与隔膜上小孔的大小有关，小孔越大，危险性越高。孔的大小有很大差别，可以是针尖大小，也可以几乎整个隔膜都缺失。如果孔很小，出生 3 年内会自动闭合。如果孔很大，左心室的血液会通过隔膜孔分流到右心室，使右心室的工作强度增加，并使肺部接收的血液过量，血压升高，这种过度的负荷会导致右心室损伤或增大。并且，肺部血流量的增多也会引起肺动脉阻塞，导致心率失常。

例 10.10　室间隔缺损的仿真

请修改 PHYSBE 的 Simulink 模型，用于模拟室间隔缺损，并预测其对于心脏血压和血容积的影响。

解：

在 PHYSBE 模型中可以将 VSD 粗略地模拟为左心室到右心室的一个分流，血流由两个心室之间的血压差驱动，可以用欧姆定律模拟，即

$$Q_s = \frac{P_{lv} - P_{rv}}{R} \quad 并且\ R = \frac{8\eta L}{\pi r^4}$$

式中　η——血粘度，值为 3cP（Fournier，1999）；

L——成年人的室间隔厚度，为 4mm（Lue 和 Takao，1986）；

Q_s——血流速率；

P_{lv}，P_{rv}——左心室和右心室的血压；

R——阻抗；

r——孔的半径。

室间隔缺损建在 PHYSBE 的左心室系统中。首先，在如图 10.34 所示的右心室系统中，用一个全局“Goto”标签将右心室血压传递给左心室系统。然后，在如图 10.35 所示的左心室系统中，用左心室血压减去右心室血压，得到血压差。再将该血压差除以阻抗，阻抗值由以下柏努利方程（Bernoulli equation）确定：

$$\frac{8 \times 3\text{cP} \times 4\text{mm}}{\pi r^4} = \frac{0.2292}{r^4}\ \frac{\text{mmHg} \cdot \text{s}}{\text{mL}}$$

其中，r 的单位为 mm。此式的倒数就是图 10.35 中的常数 C，这样，就把除法转化成了乘法。由此计算得到的分流的血流速率要从左心室血流速率中减去，并加到右心室血流速率上（即图 10.34 右心系统中的“Septal Flow”数据源模块）。

这个 VSD 模型的仿真结果显示，左心室血压到达的稳态值小于正常血压（见图 10.36），而右心室血压显著增加（见图 10.37）。这种血压变化与理论分析结果一致，因为血液从左心室分流到右心室，增加了右心室的血流量。

该仿真中设定的隔膜孔大于为 1cm^2。理论分析表明，这样大小的孔会导致大量的血液被分流到右心室中，并使得肺动脉血流量增多，引起肺循环血压上升，这种血压上升就称为高血压。

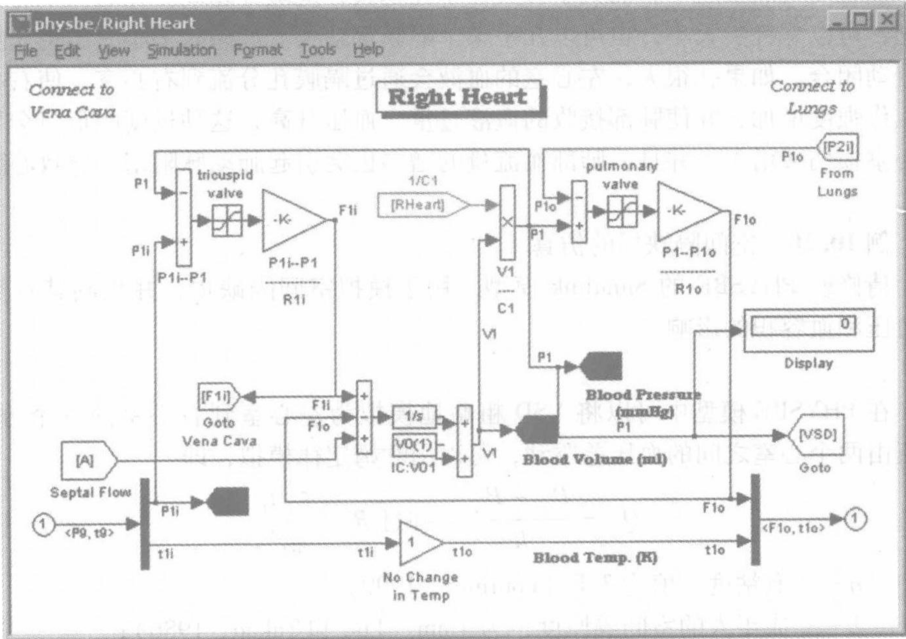


图 10.34　带室间隔缺损的 PHYSBE 右心室系统模型

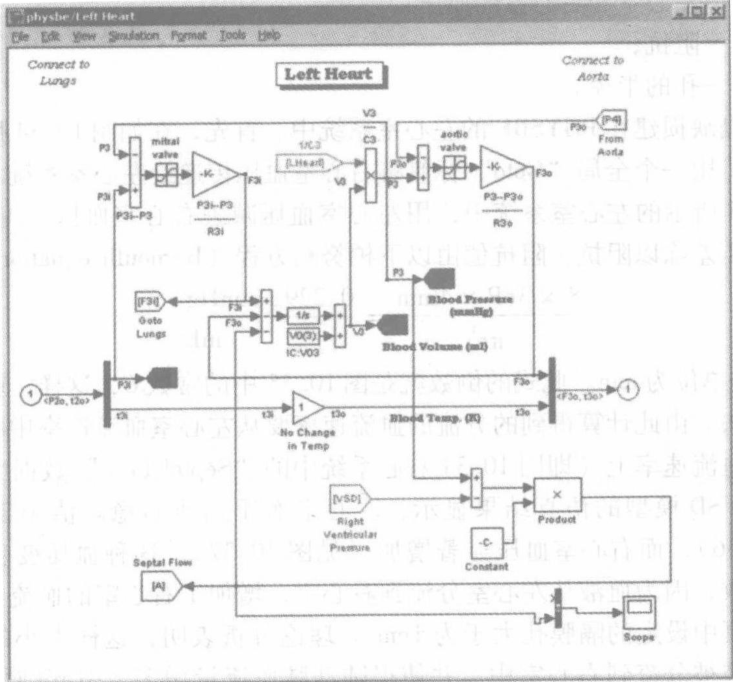


图 10.35　带室间隔缺损的 PHYSBE 左心室系统模型

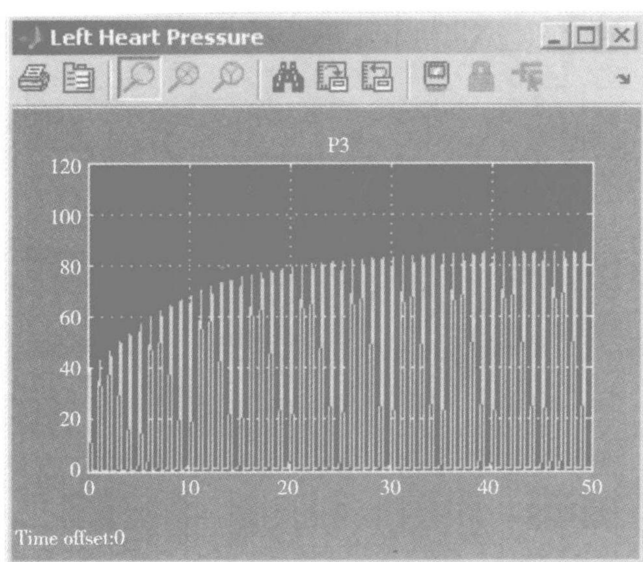


图 10.36　左心室血压

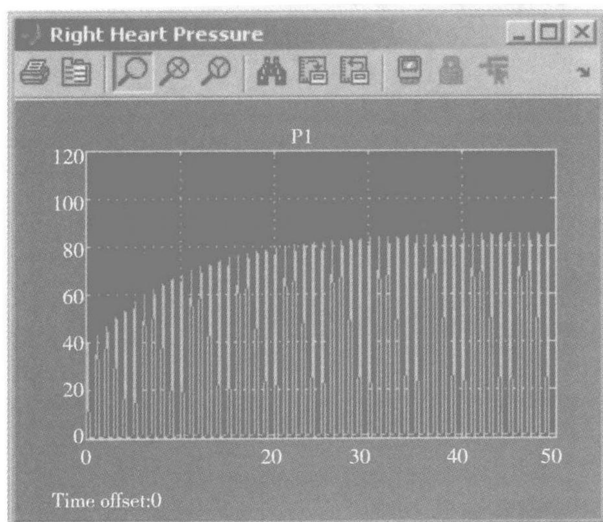


图 10.37　右心室血压

图 10.38 和图 10.39 分别显示了左、右心室的血容积，可见两个血容积都大于正常值。

室间隔缺损使左心室血液穿过隔膜流向右心室，从而影响心脏血压。如果隔膜缺损较大，在血压差驱动下大量血液分流到右心室，增加的血流量还会引起肺部血液增多，肺部血压上升，有时导致高血压。如果不及时治疗，在高血压作用

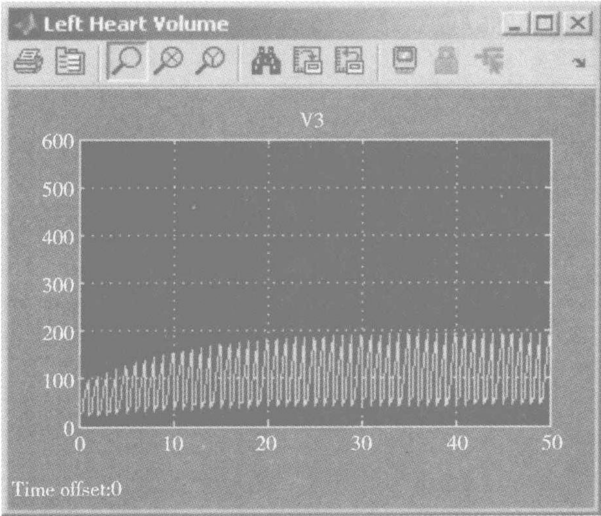


图 10.38 左心室血容积

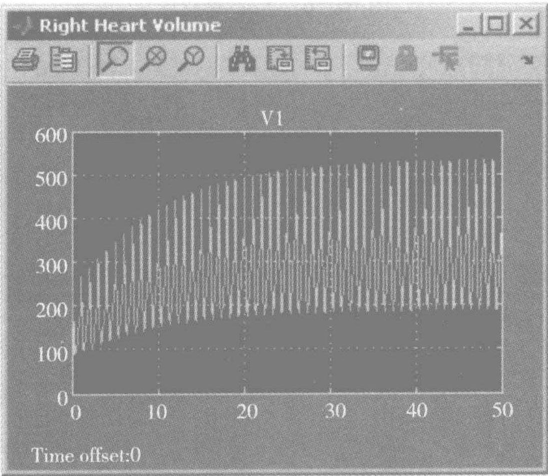


图 10.39 右心室血容积

下肺动脉会增厚，造成肺部的永久性损伤，还会导致心脏瓣膜劳损。

10.7.4 左心室肥大

长期的心肌收缩减弱会导致心室扩大，以便增加心输出量，满足人体的需求（Hori 等人，1989）。这种工作负荷的增加会导致心室壁增厚，也就是通常所称的左心室肥大（left ventricular hypertrophy）（见图 10.40）。

心室壁的内向增厚会减小舒张期的心室容积，同时也会降低心室壁的顺应性。而顺应性下降意味着心室壁的刚性增加，伴随着心室压的升高，使得左心室

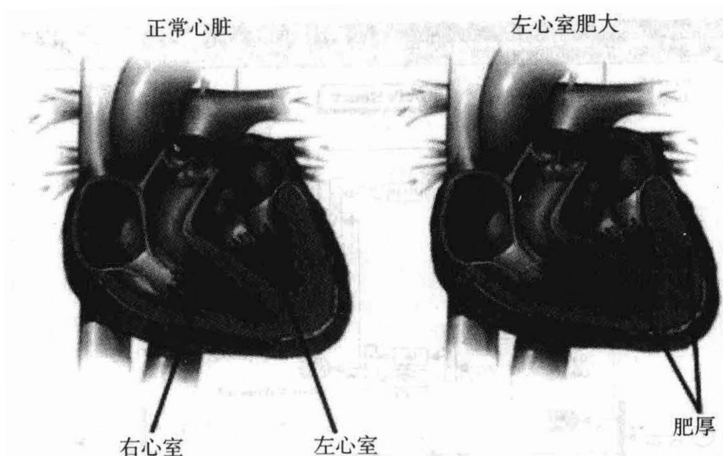


图 10.40　左心室超负荷工作引起心室壁增厚，称为左心室肥大
（照片由 Mayo 教育基金会赠送）

的容量减小。如果左心室太小，不能有效充盈，血液就会回到肺部血管，导致这些器官的血循环减少。有时，心室肥大本身是病因，但更常见的是其他疾病引起的心室肥大。

例如，慢性高血压经常是人类左心室功能障碍的起因。在正常状态下，主动脉瓣和肺动脉瓣的作用形成了血流和血循环所需的血压差。如果瓣膜工作不正常，要么发生开放障碍引起的血压过高，要么出现关闭不全引起的血容积过大(Legato，1987)。

根据欧姆定律，主动脉瓣狭窄或肺动脉瓣狭窄会使血流量减小。假设血管阻力恒定不变，两种动脉瓣的狭窄会降低血管中的血压梯度。而血压梯度的降低会引起动脉血管顺应性（$C = \Delta V/\Delta P$）的下降，反映出动脉压降低引起的血容积变化。这个模型说明，传输血管中的血量必定增加，导致左心室血容积减小。

利用 PHYSBE 系统，通过建立主动脉瓣和肺动脉瓣障碍的模型，可以间接模拟左心室肥大。因为动脉瓣障碍引起左心室血压过高，并且使流经瓣膜的血流减少。这样，就相当于发生了左心室肥大。

例 10.11　左心室肥大的仿真。

请修改 PHYSBE 的 Simulink 模型，用于模拟左心室肥大。

解：

在图 10.41 的左心室系统和图 10.42 的右心室系统中分别改变主动脉瓣和肺动脉瓣的上限值，就可以模拟动脉瓣的障碍。

除了左心室系统中的左心室血压输出示波器以外，如图 10.43 所示，要在主动脉模型中添加一个示波器，用于显示主动脉血容量的变化。

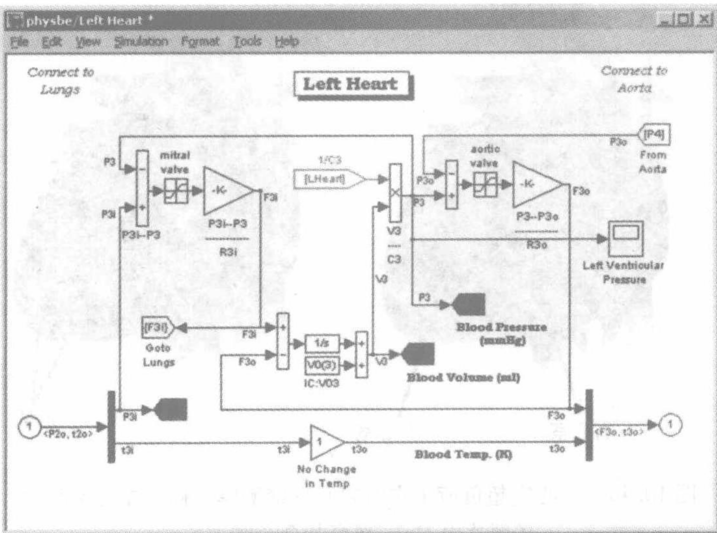


图 10.41 左心室肥大仿真系统的左心模型

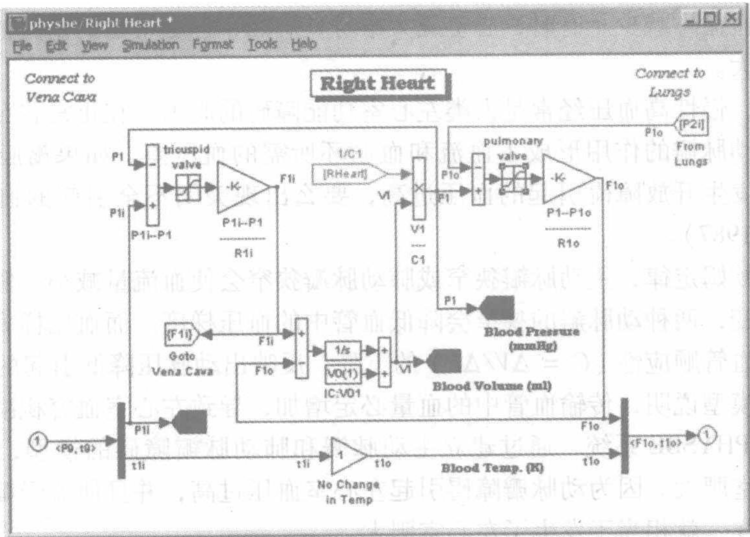


图 10.42 左心室肥大仿真系统的右心模型

这个左心室肥大模型的仿真结果显示，左心室血压上升（见图 10.44），相应地，动脉血容量减少（见图 10.45）。

左心室压力-容积（Pressure-Volume，PV）环是评价心室功能和循环系统的一种方法。

正常情况下，收缩期末压-容积关系曲线（End-Systolic Pressure-Volume Relationship，ESPVR）和舒张期末压-容积关系曲线（End-Diastolic Pressure-Volume

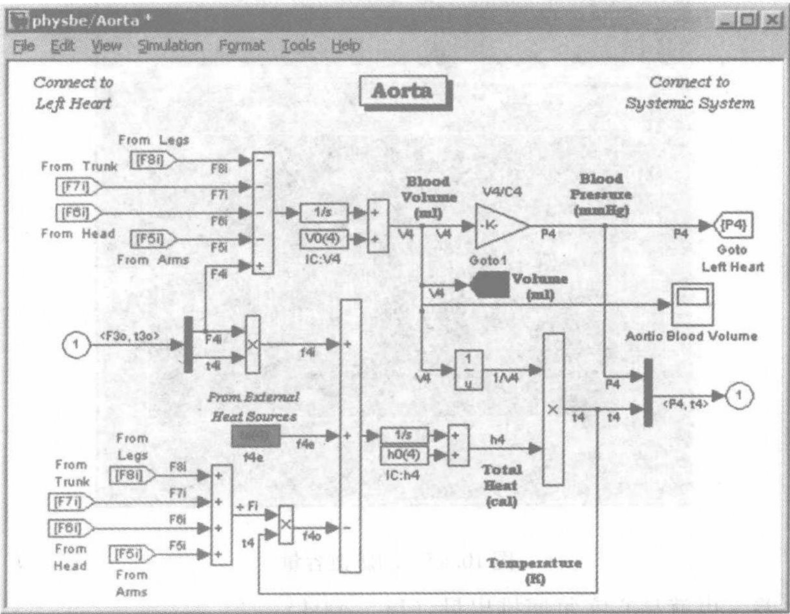


图 10.43　左心室肥大仿真系统的主动脉模型

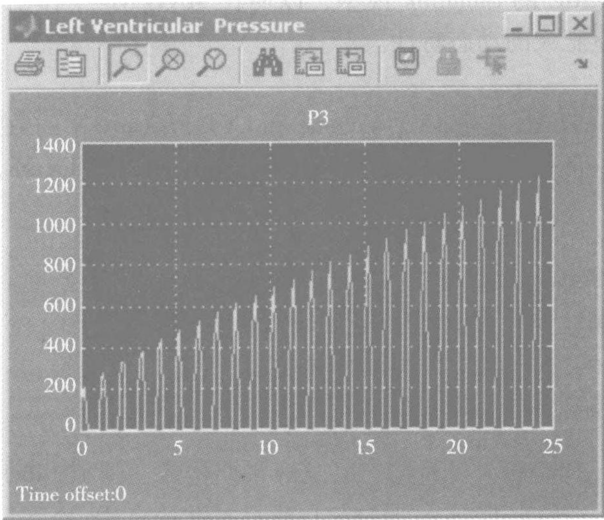


图 10.44　左心室血压

Relationship，EDPVR）确定了左心室对外做功的时间范围。ESPVR 与 PV 环相切处主动脉瓣关闭，该 EDPVR 切线与心室顺应性的倒数成正比。另外，PV 环内部的面积大小表示心室所做的机械功，PV 环的宽则等于舒张期末容积与收缩期

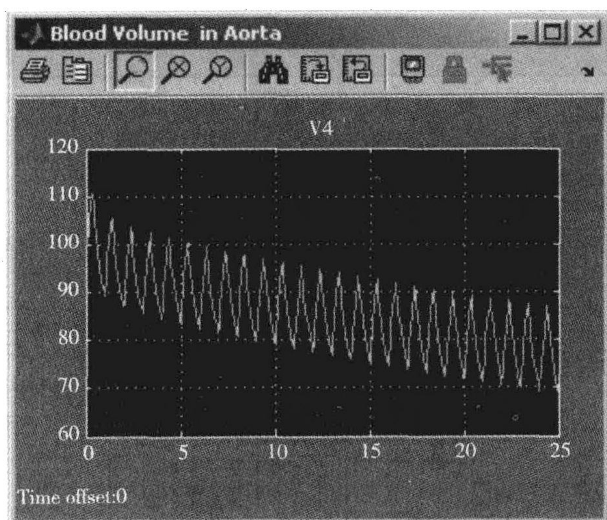


图 10.45 动脉血容量

末容积之差，也就是心室每搏排出量（Li，2004）。

例 10.12 左心室 PV 环

请修改 PHYSBE 的 Simulink 模型，以便在 MATLAB 中作出 PV 环，并利用 PV 环说明左心室肥大的动态特性。

解：

将保存在 MATLAB 变量 LV_Volume 和 LV_Pressure 中的数据分别作为横坐标和纵坐标，就可以方便地利用 PHYSBE 数据作出左心室 PV 环，MATLAB 的脚本程序如下：

```
plot(LV_Volume,LV_Pressure)
ylabel('Left Ventricular Pressure')
xlabel('Left Ventricular Volume')
axis([0  500  0  1200])
```

PV 环由充盈期、收缩期、射血期、以及舒张期构成。图 10.46 所示为正常心血管系统的 PV 环，图 10.47 所示则为左心室肥大患者的 PV 环。

正如所预料的那样，在左心室肥大的情况下，ESPVR 向右[⊖]并向上移动，PV 环也产生了移动。每个曲线环围成的总面积减小，反映了心室搏动功率和每搏排出量的减少。一旦每搏排出量减少，心率就会增加，以便维持正常的心输出

⊖ 原文中为“向左”。—译者注

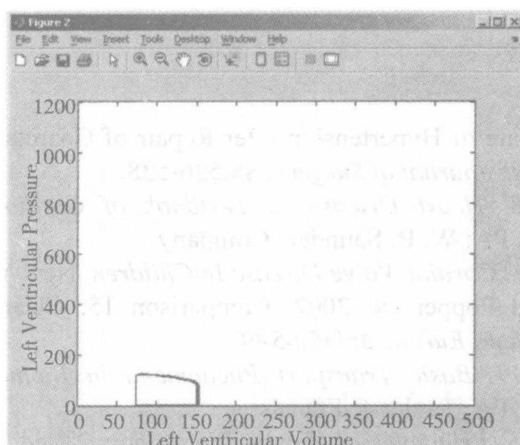


图 10.46　正常心血管系统的 PV 环

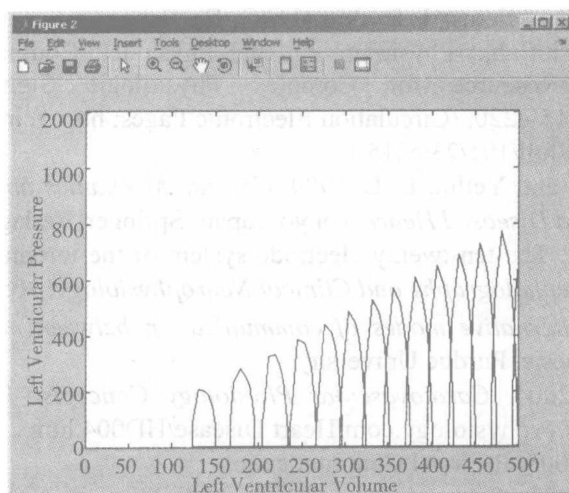


图 10.47　左心室肥大患者的 PV 环

量。这种情况如果不及时治疗，最后会导致这种适应性调节失效，使心输出量下降。

PHYSBE 模型的仿真结果显示，左心室肥大是一种补偿机制，它通过心室压的异常增加和心室容积的减小，来维持正常心输出量。并且，来自人体控制中心的刺激信号还会使心室壁的顺应性下降，刚性增加。另外，由 PV 环计算得到的每搏排出量和心搏功率的减小，就是左心室肥大的症状。

读者通过本章这些例题的学习，可以体会到数值方法在生物医学领域中具有广泛的应用，并可以学习如何使用计算工具和数值方法求解医学和生理学中出现的问题。

10.8 参考文献

Bhat, M. A. 2001. Fate of Hypertension after Repair of Coarctation of the Aorta in Adults. *British Journal of Surgery*, **88**:536-538.

Braunwald, E. 1988. *Heart Disease: a Textbook of Cardiovascular Medicine*. Philadelphia, PA: W. B. Saunders Company.

Dunn, J. M., ed. 1988. *Cardiac Valve Disease In Children*. New York: Elsevier.

Estelberger, W., and Popper, N. 2002. Comparison 15: Clearance Identification. *Simulation News Europe* **35/36**:65-66.

Fournier, R. L. 1999. *Basic Transport Phenomena in Biomedical Engineering*. Philadelphia, PA: Taylor & Francis.

Germann, W. J. 2005. *Principles of Human Physiology*, 2nd ed. San Francisco, CA: Pearson Education.

Goldberger, A. L., Amaral, L. A. N., Glass, L., Hausdorff, J. M., Ivanov, P. Ch., Mark, R. G., Mietus, J. E., Moody, G. B., Peng, C. K., and Stanley, H. E. 2000. PhysioBank, PhysioToolkit, and Physionet: Components of a New Research Resource for Complex Physiologic Signals. *Circulation* **101**(23):e215-e220. (Circulation Electronic Pages; http://circ.ahajournals.org/cgi/content/full/101/23/e215.)

Hori, M., Suga, J., and Yellin, E. L. 1989. *Cardiac Mechanics and Function in the Normal and Diseased Heart*. Tokyo, Japan: Springer-Verlag.

Jasper, H. H. 1958. The ten-twenty electrode system of the international federation. *Electroencephalography and Clinical Neurophysiology*, **10**:371-373.

Keirn, Z. 1988. *Alternative modes of communication between man and machine*. Master's thesis, Purdue University.

Klabunde, R. E. 2004. *Cardiovascular Physiology Concepts: Valvular Stenosis*. http://www.cvphysiology.com/Heart Disease/HD004.htm .

Krousgrill, C. M. 2005. Personal Communication.

Legato, M. J. 1987. *The Stressed Heart*. Boston, MA: Martinus Nijhoff Publishing.

Li, J. K-J. (2004). *Dynamics of the Vascular System*. Boston, MA: World Scientific Publishing.

Lue, H. C., and Takao A., eds. 1986. *Subpulmonic Ventricular Septal Defects*. Tokyo, Japan: Springer-Verlag.

Mascherbauer, J. 2004. Value and limitations of aortic valve resistance with particular consideration of low flow-low gradient aortic stenosis: an in vitro study. *Eur Heart J.* **25**(9):787-793.

McLeod, J. 1966. PHYSBE...a physiological simulation benchmark experiment. *SIMULATION*, **7**(6):324-329.

McLeod, J. 1968. PHYSBE...a year later, *SIMULATION*, **10**(1):37-45.

McSharry, P. E., Clifford, G. D., Tarassenko, L., and Smith, L. 2003. A dynamical

model for generating synthetic electrocardiagram signals. *IEEE Transactions on Biomedical Engineering*, **50**(3):289-294.

Mohiaddin, R. H. 1993. Magnetic Resonance Volume Flow and Jet Velocity Mapping in Aortic Coarctation. *JACC*, **22**:1515-1521.

Ruha, A., and Nissila, S. 1997. A real-time microprocessor QRS detector system with a 1-ms timing accuracy for the measurement of ambulatory HRV. *IEEE Trans Biomed Eng* **44**(3):159-167.

Sokolow, M., and McIlroy, M. B. 1981. *Clinical Cardiology*. Los Altos, CA: LANGE Medical Publications.

Suk, J., ed. 2001. Coarctation of the Aorta. *Yahoo Health*. http://health.yahoo.com/health/ency/adam/000191/i18128.

Texas Heart Institute. 2004. *Leading With the Heart*. http://www.tmc.edu/thi/vaortic.html .

Van Riel, N. 2004. *Minimal Models for Glucose and Insulin Kinetics* - A MATLAB implementation; version February, 5, 2004. Technical Report, Technische Universiteit Eindhoven.

网址：

http：//www. physionet. org/physiotools/matlab/ECGwaveGen/

http：//www. physionet. org/physiotools/ecgsyn

http：//www. emedicine. com/ped/topic2543. htm

http：//circ. ahajournals. org/cgi/content/full/101/23/e215

http：//www. cvphysiology. com/Heart Disease/HD004. htm

附　　录

附录 A　MATLAB 简介

本附录介绍 MATLAB 的基本功能，是理解本书所列举的软件程序的基础。有关这方面的内容，读者还可以参考其他 MATLAB 的自学指导书，例如 Hanselman 和 Littlefield（2005）、Hahn（2002）以及 Lyshevski（2003）等人的著作都很不错。另外，MATLAB 附带的手册也是参考资料，MATLAB 软件还带有大量详尽的在线帮助信息和教程。

读者最好坐在计算机旁，一边阅读附录，一边在 MATLAB 环境下练习每一条指令的用法。如果不实际练习运行这些指令，就不能达到预期的教学效果。符号“ >> ”是 MATLAB 指令输入的提示符，它会自动出现在指令窗中，不需要输入。本书中，以符号“ >> ”开头的行都表示 MATLAB 指令窗中的指令行。

A.1　MATLAB 环境

MATLAB7.0 版（R14）的默认工作环境有如图 A.1 所示的如下 4 个界面：

1）工作空间浏览器（Workspace）：此浏览器显示 MATLAB 程序运行期间创建和使用的变量。这些变量可以是指令窗中定义的变量，或者是调用函数和运行 M 文件时定义的变量，也可以是加载 workspace 保存文件时输入的变量。如果要查看工作空间及其变量，就打开此浏览器，或者在指令窗中用“who”和“whos”指令。如果要查看或修改变量，只要在工作空间浏览窗中双击变量名，就可以打开如图 A.2 所示的数组编辑器（Array Editor）并显示变量的值，供用户编辑修改。

2）当前目录（在图 A.1 中没有打开）：此界面显示当前目录的内容。

3）历史指令：该界面显示所有已经运行过的指令以及当前输入的指令。将鼠标指向某条指令并单击右键，出现下拉菜单，菜单上有：Copy、Evaluate Selection、Create M－file 和 Delete 选项。历史指令窗中的任意一条指令都可以被拖到指令窗重新执行。

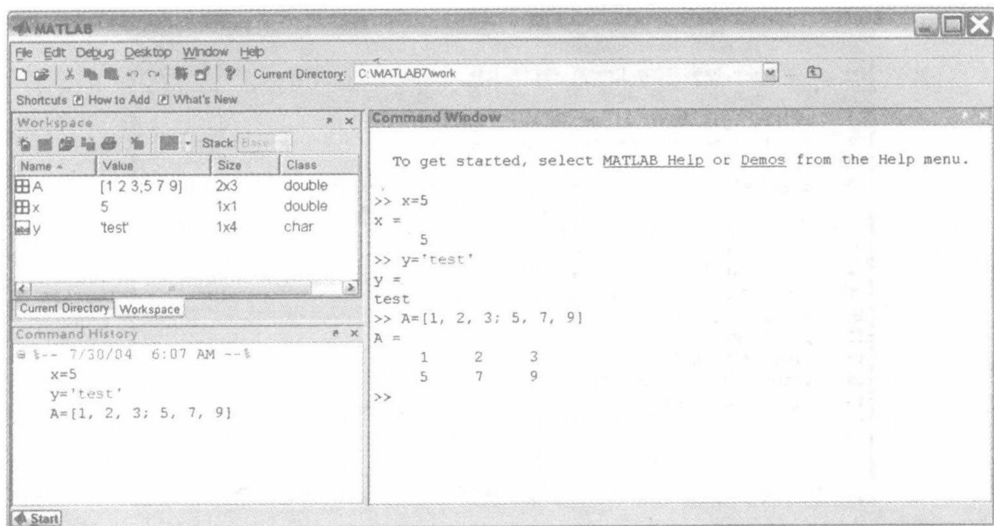


图 A.1　MATLAB 的默认工作环境

4）指令窗：此窗用于键入 MATLAB 指令，执行指令并显示输出结果。可以在该窗中输入变量、执行函数，或者运行 M 文件。

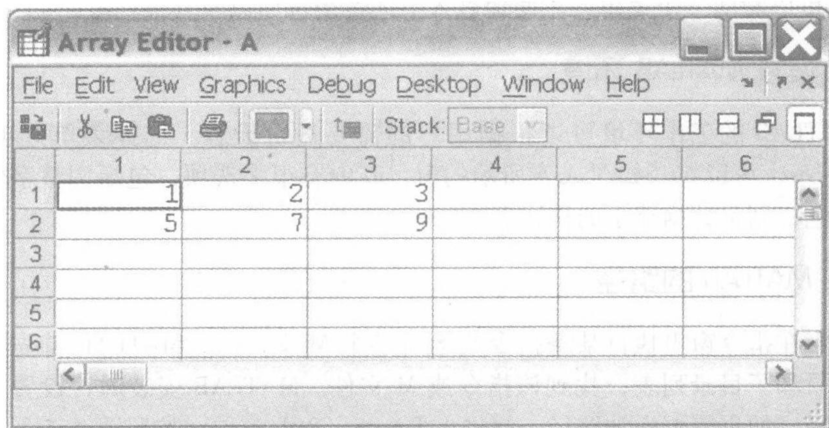


图 A.2　数组编辑器，图中所示为数组 A 的值

除了上述这些界面窗以外，还可以打开下面这些窗。

编辑窗：如图 A.3 所示是 MATLAB 编辑器，用于创建、编辑、调试和运行 MATLAB 程序（即 M 文件）。在指令窗中键入 edit 指令可以打开该 MATLAB 编辑器。

Profiler：MATLAB 有 Profiler 帮助用户改善 M 文件的运行。在 Profiler 中运行一条 MATLAB 语句或一个 M 文件，可以得到一份报告，说明程序运行的时间消

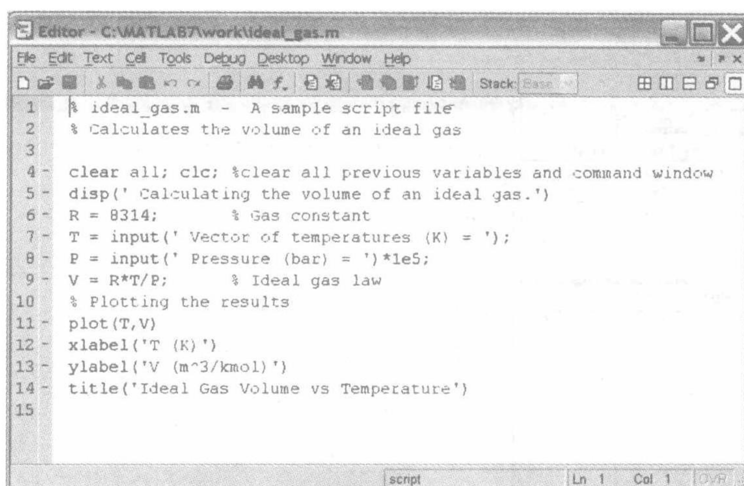


图 A. 3 MATLAB 编辑器

耗在哪里。通过桌面菜单或者调用 profile 函数可以进入 Profiler。

起始键：MATLAB 的起始键（start）位于 MATLAB 窗的左下角（见图 A. 1），可以激活一个菜单，方便地进入上述各个选项以及其他功能。

A.1.1 设置 MATLAB 环境

MATLAB 的工作环境可以根据用户的需要进行设置。选择菜单选项 File/Preferences，可以打开如图 A. 4 所示的窗，其中有很多选项，包括调整字体的大小和颜色，制定数据显示的格式等。

A.1.2 MATLAB 的路径

每当在指令窗里执行某条指令或运行某个 M 文件时，MATLAB 首先要搜索文件目录和子目录列表，找到该指令或 M 文件。MATLAB 安装时，设置了标准搜索路径。如果需要增加路径，就进入 File/Set Path 菜单，按照图 A. 5 所示的指令进行操作。

不要删除 MATLAB 标准搜索路径的文件目录，否则会影响 MATLAB 的正常运行。

命名新的 M 文件时需要谨慎，如果文件名与 MATLAB 中已经存在的文件相同，那么，MATLAB 会运行在搜索过程中首先找到的那个文件。因此，不要重复使用相同的文件名，并且，最好把用户的文件目录放在搜索路径的最后。

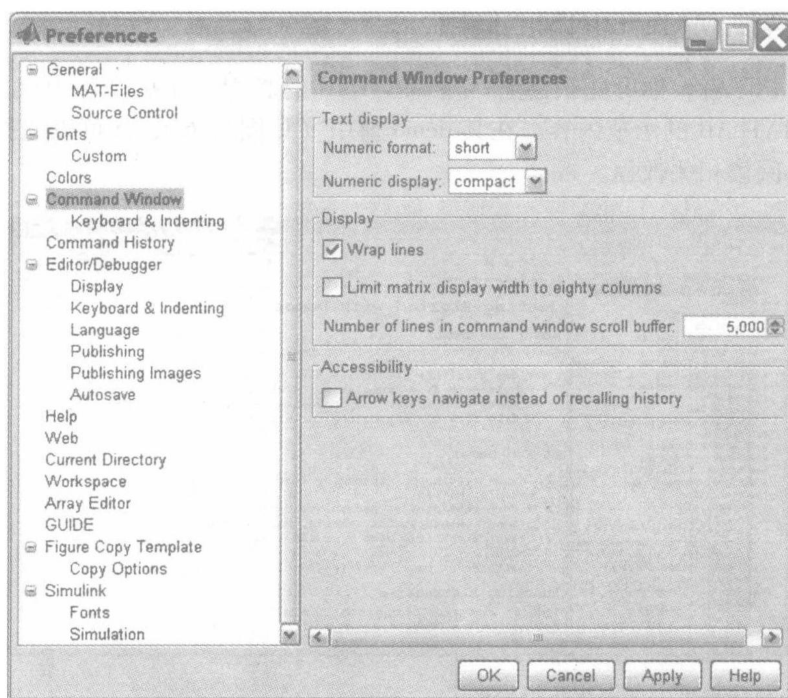


图 A. 4　设置 MATLAB 环境的 Preferences 窗

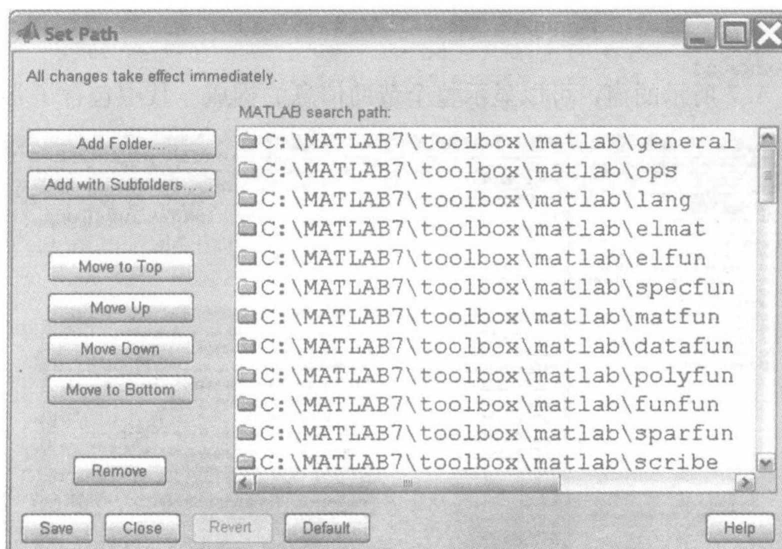


图 A. 5　MATLAB 路径的更改

A.1.3 寻找 MATLAB 的帮助信息

作为初学者，你可能很想看一下 MATLAB 的教程。在指令行中键入 demo，会显示 MATLAB 的示范内容。在其 demo 窗中（见图 A.6），你可以选择感兴趣的内容，阅读相关教程。

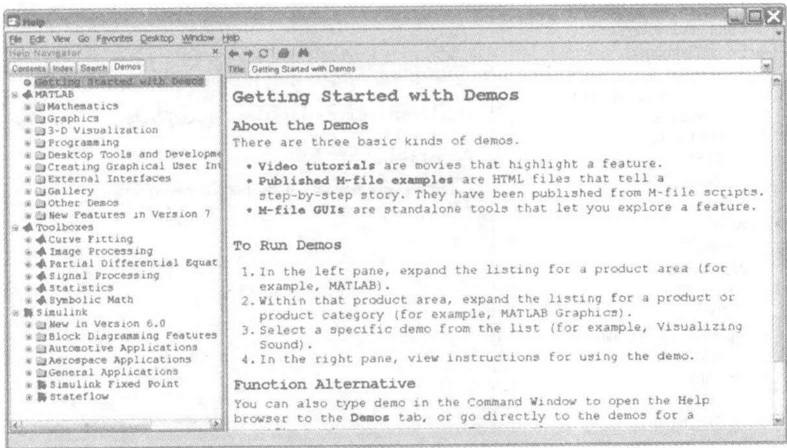


图 A.6 MATLAB 的 demo 窗

另外，使用指令

```
>> helpwin
```

打开如图 A.7 所示的窗，可以显示整个帮助信息的构成。其中包含了常用指令、

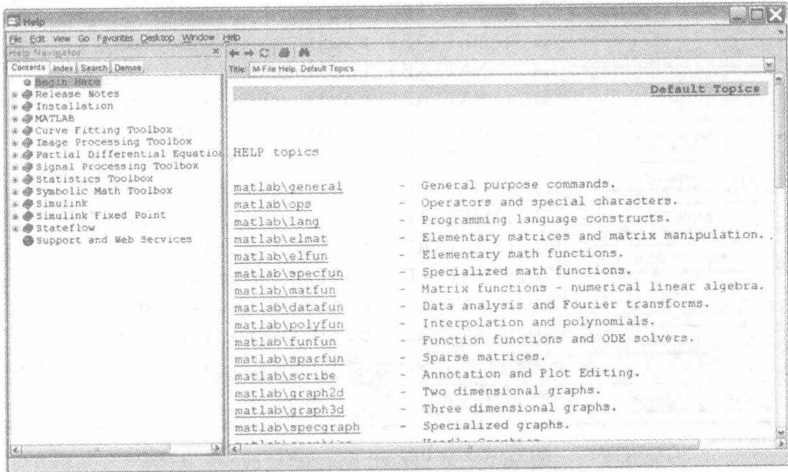


图 A.7 默认的帮助信息界面

运算符和特殊字符、基本数学函数、专用数学函数、矩阵函数等许多链接，单击链接就可以显示出所选主题的相关帮助信息。

在指令行中键入 help，会列出 MATLAB 搜索路径中的所有目录名及其说明，每个目录占一行。如果在 help 指令后跟一个目录名，那么，只要该目录在 MATLAB 的搜索路径中存在，并且包含有 content. m 文件，MATLAB 就会列出目录的内容。请测试如下指令：

```
>> help
>> help general
>> helpdesk
>> doc
```

如果函数名已知，可以用 help 指令显示其帮助信息。例如，sign 函数的帮助信息可以用如下指令显示：

```
>> help sign
SIGN  Signum function.
    For each element of X, SIGN(X) returns 1 if the element
    is greater than zero, 0 if it equals zero and -1 if it is
    less than zero. For the nonzero elements of complex X,
    SIGN(X) =X./ABS(X).
```

这些输出内容就是该函数第一段注释的内容，是函数及其功能的简单描述。MATLAB 用百分号%标记注释行，例如：

```
% This is a comment statement in a program
```

就是注释行。

如果函数名不确定，则可以用 lookfor 指令寻找，例如：

```
>> lookfor absolute
ABS      Absolute value.
IMABSDIFF Compute absolute difference of two images.
CIRCLEPICK Pick bad triangles using an absolute tolerance
MAD  Mean/median absolute deviation.
ABS     Absolute value.
LOCATEFILE Resolve a filename to an absolute location.
```

```
applyabsolute.m: % out =
applyabsolute(in, cspace, source_wp,dest_wp)
```

该指令找出了与“absolute”有关的函数。

在下列网站上可以找到广泛详尽的 MATLAB 帮助信息和使用手册：

```
http://www.mathworks.com
http://www.mathworks.com/academia/
```

A.2 基本运算符

MATLAB 的 4 个基本算术运算符，即加、减、乘、除，分别用符号“+”、“-”、“*”和“/”表示，例如：

```
>> (20 +10 -4 * 5)/2
ans =
     5
```

符号“^”表示指数运算，例如：

```
>> 5^2
ans =
    25
```

符号“\”表示左除运算，分母是在左边，而不是在右边。如下指令说明了“/”与“\”的区别：

```
>> 2/4
ans =
    0.5000
>> 2\4
ans =
     2
```

MATLAB 可以处理复数，例如：

```
>> sqrt(-1)
```

```
ans =
        0 + 1.0000i
```

也可以处理无穷大的数，例如：

```
>> 1/0
Warning: Divide by zero.
(Type "warning off MATLAB:divideByZero" to suppress this warn-
ing.)
ans =
    Inf
```

这里的 Inf 就表示无穷大。

没有新的赋值时，字母 i 和 j 原本代表复数$\sqrt{-1}$。变量 pi 则代表圆周率，即 3.141592653...。如果某个表达式不能求值，MATLAB 就返回“NaN”，意思是“不是数”（Not-a-Number），例如：

```
>> 0 * log(0)
Warning: Log of zero.
ans =
    NaN
```

MATLAB 中的等号表示给变量赋值，如：

```
>> a =2
a =
     2
>> b =3 * a
b =
     6
```

如果没有把计算结果赋值给变量，那么表达式的值就存放在默认变量 ans 中，例如：

```
>> a +b
ans =
     8
```

如果不想在指令窗中显示出赋值的结果，在赋值语句末尾加上分号就可以了：

```
>> c = a + b;
```

直接键入变量名可以查看变量的值，例如：

```
>> c
c =
      8
```

MATLAB 区分大小写字母，也就是大写字母和小写字母代表不同的变量名，例如：

```
>> A = 10;
>> A
A =
      10
>> a
a =
       2
```

所有的 MATLAB 运算都使用双精度（见第 3 章），计算的精度与数据的显示格式无关，计算结果一般只显示 5 位有效数字。利用 format 指令可以改变显示格式，例如：

```
>> d = exp(pi)
d =
    23.1407
>> format long, d
d =
   23.14069263277927
>> format short e, d
d =
   2.3141e + 001
>> format long e, d
d =
    2.314069263277927e + 001
```

```
>> format short, d
d =
   23.1407
```

用 help 指令可以查看 format 指令的各种不同设置参数:

```
>> help format

FORMAT Set output format.
   All computations in MATLAB are done in double precision.
   FORMAT may be used to switch between different output
   display formats as follows:
     FORMAT          Default. Same as SHORT.
     FORMAT SHORT    Scaled fixed point format with 5 digits.
     FORMAT LONG     Scaled fixed point format with 15 digits.
     FORMAT SHORT E  Floating point format with 5 digits.
     FORMAT LONG E   Floating point format with 15 digits.
     FORMAT SHORT G Best of fixed or floating point format with 5
digits.
     FORMAT LONG G Best of fixed or floating point format with 15
digits.
     FORMAT HEX      Hexadecimal format.
     FORMAT +        The symbols +, - and blank are printed
                   for positive, negative and zero elements.
                   Imaginary parts are ignored.
     FORMAT BANK   Fixed format for dollars and cents.
     FORMAT RAT   Approximation by ratio of small integers.

   Spacing:
     FORMAT COMPACT Suppress extra line-feeds.
     FORMAT LOOSE   Puts the extra line-feeds back in.
```

用 clear 指令可以从内存中删除某个变量:

```
>> clear a
```

用 clear 或 clear all 指令则删除 MATLAB 工作空间 (Workspace) 中的所有变量:

```
>> clear 或者 >> clear all
```

请查看工作空间窗，确认该指令运行之后，其中的所有变量已被删除。

指令 clc 清除指令窗中所显示的内容，并把光标置于窗的顶部：

```
>> clc
```

指令 clf 清除当前的图形窗：

```
>> clf
```

记住用“↑”键和“↓”键可以滚动显示已经键入的所有指令。如果需要调出已经使用过的某条指令，只要键入其第一个字母，或者前几个字母，然后用“↑”键找出指令。

在 MATLAB 指令窗中还可以执行操作系统的目录操作指令，例如：cd（设置当前目录）、dir（列出当前目录中的文件）、mkdir（创建新目录）、pwd（显示当前工作目录）、ls（目录列表）。

例如，以下指令将 MATLAB 的工作目录设置为当前目录：

```
>> cd c:\matlab704\work
```

将生物系统程序目录中的第一章目录设置为当前目录，则为

```
>> cd 'C:\Program Files\Biosystems\Chapter1'
```

由于该目录名中存在空格，因此，指令中需要用单引号“'”。由此可见，最好不要在目录名和 M 文件名中使用空格，如果要分开文件名中的单词，应该用下划线“_”。

A.3 矢量和矩阵

MATLAB 这个名称是矩阵实验室（Matrix Laboratory）的缩写，它是一个专门用于矩阵计算的软件。MATLAB 的变量都用数组表示，标量是 1×1 数组，行矢量是 $1 \times n$ 数组，列矢量则是 $n \times 1$ 数组。MATLAB 的二维数组用标准矩阵符号 M（行，列）表示。也可以用大于二维的多维矩阵，例如，M（行，列，页），第三维就是页，但是，再高的维数就没有通用的名称了。

给 MATLAB 变量赋值就可以创建一个矩阵，此时矩阵的元素必须放在方括号中，即

```
>> M=[1,2,3;4,5,6]
M=
    1    2    3
    4    5    6
```

或者

```
>> M=[1,2,3
      4 5 6]
M=
    1    2    3
    4    5    6
```

每行的元素之间可以用逗号或者空格隔开，行与行之间用分号或者回车（即换行）分开。矩阵中的某个元素用 M（行，列）表示，可以对其进行单独存取和置换，例如：

```
>> M(1,3)
ans =
     3
>> M(2,1)=7
M=
    1    2    3
    7    5    6
```

其中的“行”和“列”是两个变量，用于指示元素在矩阵中的位置。矩阵的下标必须是大于或者等于 1 的整数，MATLAB 不允许用 0 作为矩阵的下标。

两个矩阵可以联合，形成一个新的矩阵，例如：

```
>> N=[M;M]
N=
    1    2    3
    7    5    6
    1    2    3
    7    5    6
>> O=[M,M]
O=
```

```
    1    2    3    1    2    3
    7    5    6    7    5    6
```

矩阵的行列交换是矩阵的转置运算，用矩阵名之后加上单引号表示，即：

```
>> M_trans = M'
M_trans =
     1    7
     2    5
     3    6
```

MATLAB 有一个方便的速写方法用于表示数的序列，就是用冒号“:”运算符，两个数之间加上冒号，表示一个数的矢量，其值从左边界值递增到右边界值，默认的递增增量为 1，例如：

```
>> v = [-1:4]
v =
    -1    0    1    2    3    4
```

但是，用户可以随意改变递增增量，例如：

```
>> W = [-1:0.5:2; 6:-1:0; 1:7]
W =
  -1.0000  -0.5000        0   0.5000   1.0000   1.5000   2.0000
   6.0000   5.0000   4.0000   3.0000   2.0000   1.0000        0
   1.0000   2.0000   3.0000   4.0000   5.0000   6.0000   7.0000
```

通常用冒号运算符来引用矩阵的行、列、或者其中某个部分，例如：

```
>> W(:, 5)   % 该指令引用第 5 列的所有行
ans =
     1
     2
     5
>> W(1, :)   % 这是引用第 1 行的所有列
ans =
 -1.0000  -0.5000  0  0.5000  1.0000  1.5000  2.0000
```

```
>> W(2:3, 4:7)   % 这是引用第 2 行到第 3 行的第 4 列到第 7 列
ans =
      3  2  1  0
      4  5  6  7
>> W(2, 5:end)   % 这是引用第 2 行的第 5 列到最后一列
ans =
      2  1  0
```

MATLAB 也可以处理高于二维的多维矩阵，例如，下面的指令是在二维矩阵 W 中增加一维，使其变成三维矩阵：

```
>> W(:,:,2)=[2:2:14; 0.1 0.2 0.3 0.4 0.5 0.6 0.7; 2 2 2 2 2 2 2]
W(:,:,1) =
  -1.0000  -0.5000        0   0.5000   1.0000   1.0000   2.0000
   6.0000   5.0000   4.0000   3.0000   2.0000   1.0000        0
   1.0000   2.0000   3.0000   4.0000   5.0000   6.0000   7.0000
W(:,:,2) =
   2.0000   4.0000   6.0000   8.0000  10.0000  12.0000  14.0000
   0.1000   0.2000   0.3000   0.4000   0.5000   0.6000   0.7000
   2.0000   2.0000   2.0000   2.0000   2.0000   2.0000   2.0000
```

前面已经提到过，第三维称为页。

A.3.1 创建特殊数组的 MATLAB 函数

MATLAB 有许多内置的数组创建函数，例如：

```
>> ones(2)          % 生成一个元素值均为 1 的(2×2)矩阵
ans =
      1    1
      1    1
>> ones(2,3)        % 生成一个元素值均为 1 的(2×3)矩阵
ans =
      1    1    1
```

```
        1     1     1
 >> zeros(2,3,2)   % 生成一个元素值均为 0 的(2×3×2)数组
ans(:,:,1) =
        0     0     0
        0     0     0
ans(:,:,2) =
        0     0     0
        0     0     0
 >> eye(3)           % 生成一个(3×3)单位矩阵
ans =
        1     0     0
        0     1     0
        0     0     1
 >> rand(4,2)        % 生成一个(4×2)的随机矩阵
                     % 其元素值均匀分布于 0.0~1.0 之间
ans =
       0.9501    0.8913
       0.2311    0.7621
       0.6068    0.4565
       0.4860    0.0185
 >> linspace(-1,5,7)  % 生成一个包含 7 个元素的行矢量
                      % 元素的值在 -1~5 之间等间隔取值
ans =
      -1     0     1     2     3     4     5
 >> logspace(-1,2,8)  % 生成一个包含 8 个元素的对数等分行矢量
                      % 元素的值在 10^-1~10^2 之间根据对数等分取值
ans =
      0.1000    0.2683    0.7197    1.9307    5.1795    13.8950
37.2759    100.0000
```

“size” 和 “length” 是两个很有用的数组函数，分别用于求取数组各维的长度和最大长度，例如：

```
 >> size(W)
ans =
       3     7     2
```

```
>> length(W)
ans =
     7
```

A.3.2　数组的算术运算

数组与标量相乘等于数组中的每个元素与该标量相乘，例如：

```
>> A=[1,2,6;2:4;10:0.5:11],2*A
A =

  1.0000   2.0000   6.0000

  2.0000   3.0000   4.0000

 10.0000  10.5000  11.0000

ans =

     2      4     12
     4      6      8
    20     21     22
```

同维同大小的数组才能进行加减运算，例如：

```
>> B=[4,7,14;1,5,7;7,7,5]
B =

     4      7     14
     1      5      7
     7      7      5

>> A-B
ans =

  -3.0000  -5.0000  -8.0000

   1.0000  -2.0000  -3.0000

   3.0000   3.5000   6.0000

>> c=[5;2;4];
>> A+c
```

```
??? Error using = = > +
Matrix dimensions must agree.
```

数组与标量相加等于数组中的每个元素与该标量相加，例如：

```
>> A+5
ans =

    6.0000    7.0000   11.0000

    7.0000    8.0000    9.0000

   15.0000   15.5000   16.0000
```

矩阵之间相乘需要满足一定的条件（参见附录C），例如：

```
>> A*B              % 可相乘,因为两个矩阵都是(3×3)
ans =

    48.0000    59.0000    58.0000

    39.0000    57.0000    69.0000

   127.5000   199.5000   268.5000

>> A*c              % 可相乘,因为A是(3×3),c是(3×1)
ans =

     33

     32

    115

>> c*A              % 不可相乘,因为c是(3×1),而A是(3×3)
??? Error using = = > *
Inner matrix dimensions must agree.

>> c'*A             % 可相乘,因为c'是(1×3),A是(3×3)
ans =
    49    58    82
```

在运算符前加上“.”，表示进行矩阵中对应元素之间的运算，例如：

```
>> A.*B
ans =

    4.0000   14.0000   84.0000

    2.0000   15.0000   28.0000

   70.0000   73.5000   55.0000

>> A.^2
ans =

    1.0000      4.0000     36.0000

    4.0000      9.0000     16.0000

  100.0000    110.2500    121.0000

>> A^2
ans =
     65       71       80
     48       55       68
    141      167      223
>> 1./A
ans =

    1.0000    0.5000    0.1667

    0.5000    0.3333    0.2500

    0.1000    0.0952    0.0909
```

下面这些矩阵函数很有用:

```
>> det(A)              % 求方阵的行列式值
ans =
   -27
>> inv(A)              % 求方阵的逆
ans =

    0.3333   -1.5185    0.3704

   -0.6667    1.8148   -0.2963
```

```
    0.3333  -0.3519   0.0370
>> rank(A)           % 求矩阵的秩(参见附录C)
ans =
     3
>> [V,D]=eig(A)   % 求方阵的特征矢量和特征值
V =

   -0.3457   -0.7603    0.6014

   -0.2817   -0.1527   -0.7737

   -0.8951    0.6314    0.1993

D =

   18.1660         0         0

         0   -3.5811         0

         0         0    0.4150
```

其中，矩阵 V 的列就是矩阵 A 的特征矢量，矩阵 D 的对角线元素就是与这些特征矢量相对应的特征值。

```
>> poly(A)           % 矩阵A的特征多项式的系数
ans =
    1.0000  -15.0000  -59.0000  27.0000
```

这是指矩阵 A 的特征多项式（即 det（A－λI））为

$$\lambda^3 - 15\lambda^2 - 59\lambda + 27$$

A.4 MATLAB 的内置函数

MATLAB 既有基本的也有高级的标准数学运算函数和可视化作图函数，例如：abs、sqrt、exp、sin、cos、eig、plot、plot3 等。如下 help 指令可用于查看基本函数（elementary functions）列表：

```
>> help elfun
  Elementary math functions.
```

```
Trigonometric.
  sin      -Sine.
  sind     -Sine of argument in degrees.
  sinh     -Hyperbolic sine.
  asin     -Inverse sine.
  asind    -Inverse sine, result in degrees.
  asinh    -Inverse hyperbolic sine.
  cos      -Cosine.
  cosd     -Cosine of argument in degrees.
  cosh     -Hyperbolic cosine.
  acos     -Inverse cosine.
  acosd    -Inverse cosine, result in degrees.
  acosh    -Inverse hyperbolic cosine.
  tan      -Tangent.
  tand     -Tangent of argument in degrees.
  tanh     -Hyperbolic tangent.
  atan     -Inverse tangent.
  atand    -Inverse tangent, result in degrees.
  atan2    -Four quadrant inverse tangent.
  atanh    -Inverse hyperbolic tangent.
  sec      -Secant.
  secd     -Secant of argument in degrees.
  sech     -Hyperbolic secant.
  asec     -Inverse secant.
  asecd    -Inverse secant, result in degrees.
  asech    -Inverse hyperbolic secant.
  csc      -Cosecant.
  cscd     -Cosecant of argument in degrees.
  csch     -Hyperbolic cosecant.
  acsc     -Inverse cosecant.
  acscd    -Inverse cosecant, result in degrees.
  acsch    -Inverse hyperbolic cosecant.
  cot      -Cotangent.
  cotd     -Cotangent of argument in degrees.
  coth     -Hyperbolic cotangent.
```

```
  acot      -Inverse cotangent.
  acotd     -Inverse cotangent, result in degrees.
  acoth     -Inverse hyperbolic cotangent.

Exponential.
  exp       -Exponential.
  expm1     -Compute exp(x) - 1 accurately.
  log       -Natural logarithm.
  log1p     -Compute log(1 + x) accurately.
  log10     -Common (base 10) logarithm.
  log2      -Base 2 logarithm and dissect floating point number.
  pow2      -Base 2 power and scale floating point number.
  realpow   -Power that will error out on complex result.
  reallog   -Natural logarithm of real number.
  realsqrt  -Square root of number greater than or equal to zero.
  sqrt      -Square root.
  nthroot   -Real n-th root of real numbers.
  nextpow2  -Next higher power of 2.

Complex.
  abs       -Absolute value.
  angle     -Phase angle.
  complex   -Construct complex data from real and imaginary parts.
  conj      -Complex conjugate.
  imag      -Complex imaginary part.
  real      -Complex real part.
  unwrap    -Unwrap phase angle.
  isreal    -True for real array.
  cplxpair  -Sort numbers into complex conjugate pairs.

Rounding and remainder.
  fix       -Round towards zero.
  floor     -Round towards minus infinity.
  ceil      -Round towards plus infinity.
  round     -Round towards nearest integer.
```

```
mod       -Modulus (signed remainder after division).
rem       -Remainder after division.
sign      -Signum.
```

以下 help 指令可以查看特殊的数学函数列表：

```
>> help specfun
```

用以下 help 指令查看矩阵及其操作的基本函数列表：

```
>> help elmat
```

查看图形函数列表，则用以下指令：

```
>> help graphics
```

查看运算符和特殊字符列表，用以下指令：

```
>> help ops
```

如果要查看工具箱（Toolboxs）的帮助信息，则在 help 指令之后加上工具箱的名称即可，例如：

```
>> help symbolic math 或 >> help simulink
```

A. 5　图形

MATLAB 有很多不同的可视化工具可以用于显示数据和函数曲线，例如：二维（2-D）和三维（3-D）图形、圆形百分比图、柱状图、直方图和轮廓图等。

1. 二维图形

如图 A. 8 所示，使用如下 MATLAB 指令可以方便地显示单变量函数曲线：

```
>> x = linspace(0,2,30);          % 在 0 ~2 之间等间隔取值
                                  % 创建一个具有 30 个元素的矢量,
>> y = x. * exp( - x);            % 计算对应于矢量 x 的 y 函数值
>> plot(x,y)                      % 以 x 为横坐标,y 为纵坐标作图
>> grid                           % 在坐标系中添加网格线
>> xlabel('Distance')             % 标注 x 坐标
```

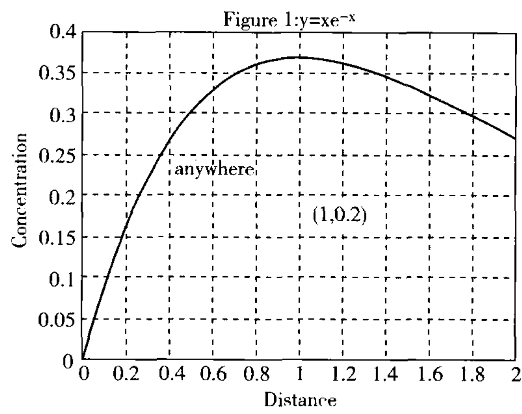


图 A.8 二维图形示例

```
>> ylabel('Concentration')          % 标注 y 坐标
>> title('Figure 1: y=xe^-^x')      % 添加图的标题
>> gtext('anywhere')                % 添加文本,并用鼠标确定其位置
>> text(1,0.2,'(1,0.2)')            % 在指定位置上添加文本
```

函数曲线也可以用符号来表示，而不用画线，如图 A.9 所示，还可以把多

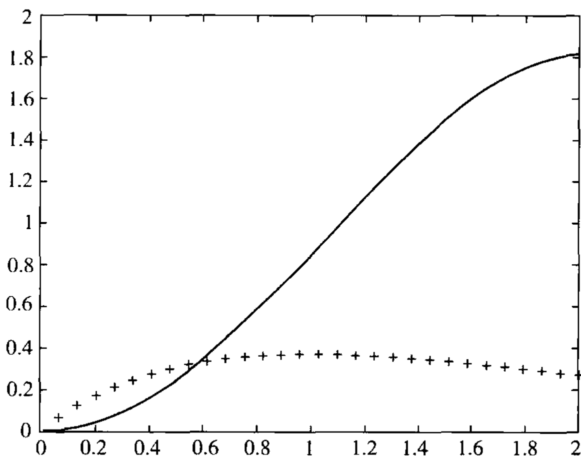


图 A.9 在同一张图上作出多个函数曲线

个函数画在同一张图上，指令如下：

```
>> plot(x, y, '+', x, x.*sin(x))
```

根据需要，一张图还可以分成多个子图。如图 A.10 所示，以下指令创建了一个 (2×1) 的图形阵列，并且，把一个函数曲线放在子图 1 中，把另一个函数曲线

放在子图 2 中。

```
>> subplot(2, 1, 1), plot(x, x.*cos(x))
>> subplot(2, 1, 2), plot(x, x.*sin(x))
```

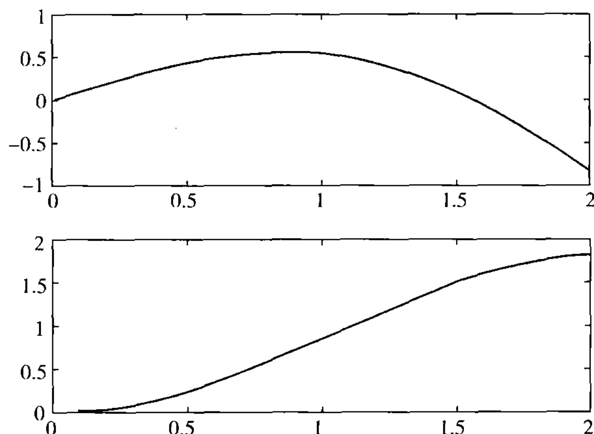


图 A.10　在一张图上作出多个子图

利用“axis”指令，可以查看或者改变坐标值的取值范围，例如：

```
>> axis
ans =
      0    2    0    2
```

请输入如下指令，并观察图形的变化：

```
>> axis([0, 1.5, 0, 1.5])
```

现在，清除图形窗：

```
>> clf
```

再用如下指令来看一下慧星轨迹函数图：

```
>> shg, comet(x,y)
```

其中，“shg”指令用于显示当前图形窗。如果用“figure (n)”指令则可以同时开设并使用多个图形窗，这里的 n 为正整数。

“fplot (FUN, LIMS)”也是一种简单的函数作图方法，它在 LIMS = [XMIN

XMAX] 所限定的 x 坐标范围内，作出函数 FUN 的图，例如：

```
>> figure(2)                    % 创建第二个图形窗
>> fplot('x * exp( - x)',[0, 2])
```

此处的函数也可以是用户自定义函数（参见 A.6 节）。

读者最好也练习一下其他的二维作图功能，例如：

```
>> plotyy(x1,y1,x2,y2)  % 双纵坐标图,显示 x1 与左边 y 轴表示的 y1 
配对的曲线,
                        % 以及 x2 与右边 y 轴表示的 y2 配对的曲线
>> semilogx(x,y)        % 半对数坐标图,其中 x 轴为对数坐标
>> semilogy(x,y)        % 半对数坐标图,其中 y 轴为对数坐标
>> loglog(x,y)          % 全对数二维坐标图,两个坐标均为对数
>> area(x,y)            % 面域图
>> polar(x,y)           % 极坐标曲线图
>> bar(x,y)             % 柱状图
```

2. 三维图形

MATLAB 有一些三维函数的作图指令，如下“plot3”指令可以显示三维曲线（见图 A.11）：

```
>> t =0:0.01:3 * pi;
>> plot3(t, sin(t), cos(t))
>> xlabel('t'), ylabel('sin t'), zlabel('cos t')
```

三维表面（见图 A.12）可以有多种显示方法，下面是两个例子：

```
>> [x, y] =meshgrid( - pi:pi/10:pi, - pi:pi/10:pi);
>> z =cos(x). * cos(y);
>> surf (x, y, z)
```

另一种显示方法是：

```
>> mesh(x, y, z)
>> view(30,60)
```

再用“shading”指令可以使图形更漂亮：

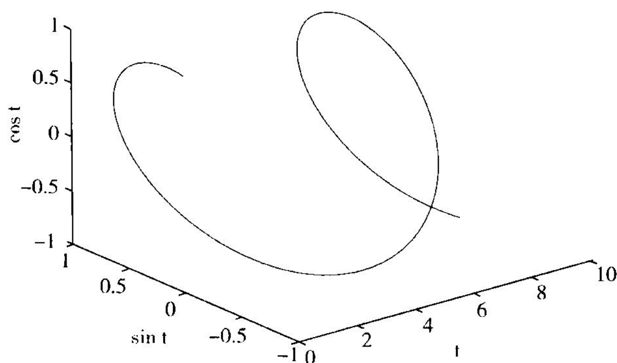


图 A. 11　三维曲线图示例

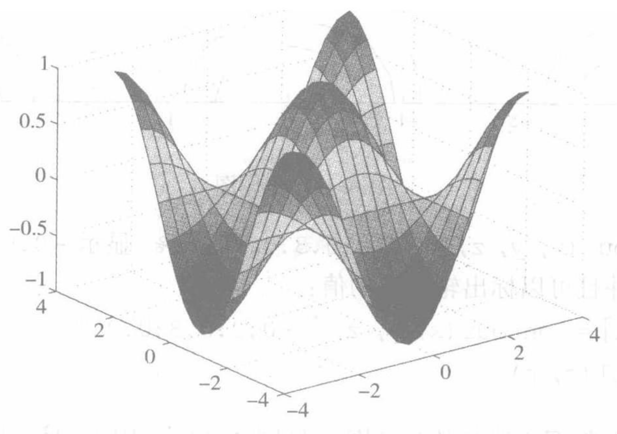


图 A. 12　三维表面图示例

```
>> shading interp
```

用 “colorbar” 指令还可以显示色条：

```
>> colorbar
```

3. 2½-D 图形

所谓的 2½-D 图是指在二维坐标系统中显示三维图形。这里，我们就用前面的 x、y 和 z 数据来作 2½-D 图，如图 A. 13 是用轮廓线在 x-y 坐标系统中显示不同的 z 值水平，指令为

```
>> contour(x, y, z)
```

也可以根据需要，选择显示几条特定的轮廓线：

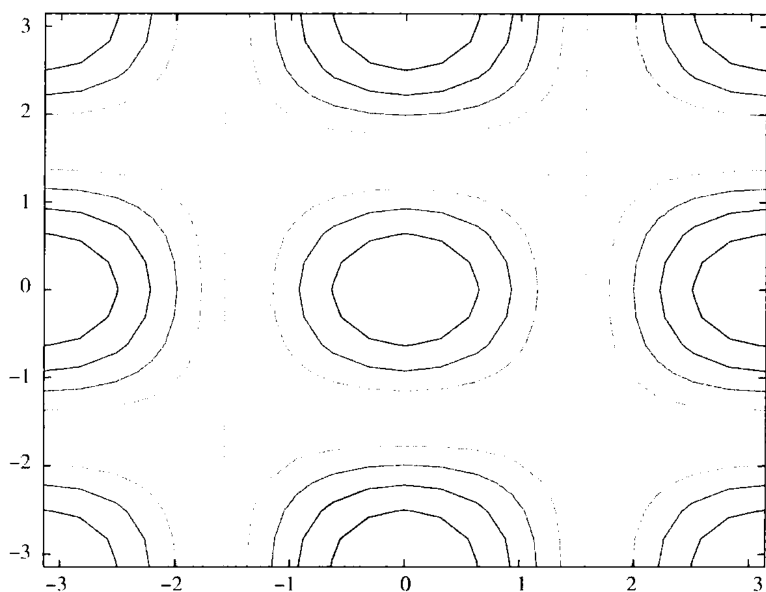


图 A.13 轮廓图

```
>> contour(x, y, z, [-0.9:0.3:0.9])  % 显示 -0.9 ~ 0.9 之间间隔 0.3 的轮廓线并且可以标出轮廓线的值:
>> [c, h] = contour(x, y, z, [-0.9:0.3:0.9]);
>> clabel(c, h)
```

还有一种 2½-D 作图方法是伪彩色图（见图 A.14），用不同的颜色表示不同的 z 值，指令如下：

```
>> pcolor(x, y, z)
>> colorbar
>> shading interp
```

此外，“quiver” 指令对于表示速度等矢量场很有用。

利用 MATLAB 图形窗口的菜单，可以对图形进行编辑、旋转、缩放和修改，下一节讲述这些功能。

4. 交互式图形的创建

MATLAB 7 引进了一组新的作图工具，用户无需编写 MATLAB 程序，直接选取数据，就可以快速创建并编辑图形，将数据可视化。还可以自动生成相应的作图程序，便于将此程序用于其他数据的作图显示。如图 A.15 所示，交互式图

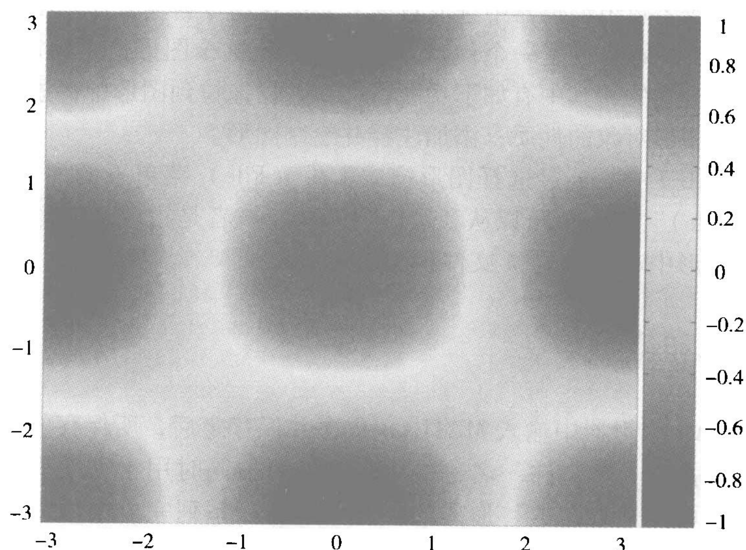


图 A. 14　伪彩色图

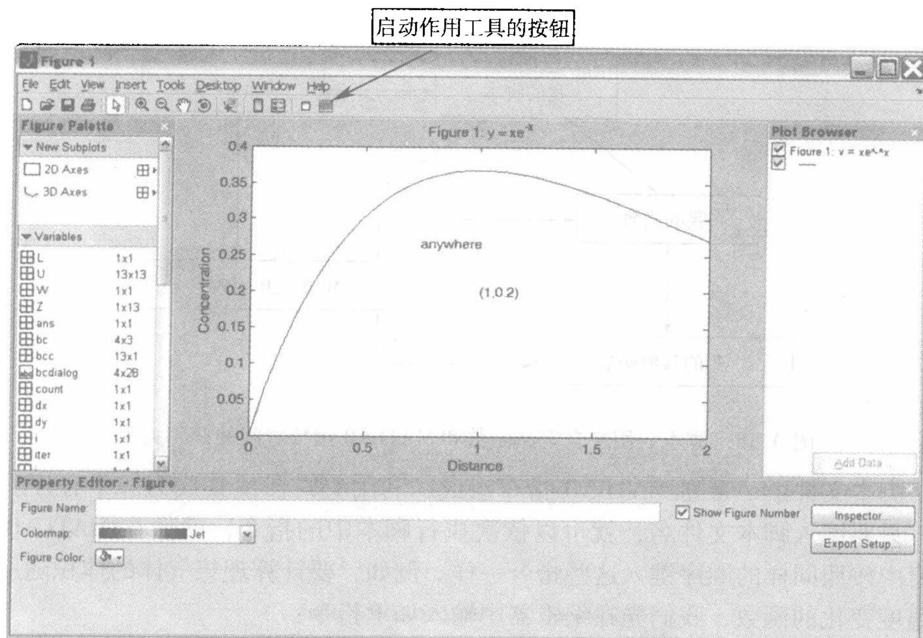


图 A. 15　点击图中指示的按钮可以打开作图工具

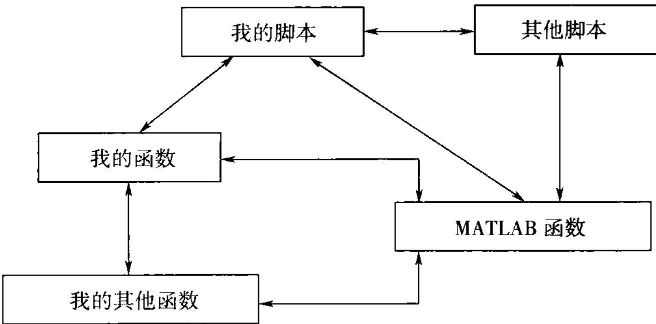
形的创建和编辑通过作图工具实现，从图形窗的工具栏可以启动此作图工具。可以通过新开空白图形窗来启用作图工具，但是通常不这么做。用户经常要么在工

作空间浏览器或者数组编辑器中选取数据、要么在指令窗输入作图函数、或者打开现有的.fig文件，来创建一个初始图形，然后从这个图形窗中启动作图工具。作图工具是点击式的，其中有如下功能：将新数据拖拉到图形中；添加新的子图坐标轴；改变图形对象的性质；添加注释并绘制图形。

在MATLAB 7中，通过选择图形窗中文件（File）菜单下的“生成M文件（Generate M-file）”选项，可以从利用作图工具（或者是指令行中键入的作图函数）所作的图形中自动生成M文件程序。

A.6 脚本和函数

在MATLAB编辑器中输入MATLAB程序并保存之后，所保存的文件都带有扩展名m，这就是“M文件”名称的由来。编辑器同时用于这类程序的调试和运行。M文件可以是“脚本”形式的，也就是由一系列完成某项任务的指令组成，其功能与主程序以及函数类似，可用于反复执行各种计算等任务。程序的调试一旦通过，M脚本文件就可以在MATLAB中运行，并通过调用用户自己编写的函数或者MATLAB的内置函数，完成某些任务，然后返回结果。如图A.16所示，脚本文件也可以调用别的脚本。



图A.16 脚本、用户自定义函数和MATLAB函数之间的调用关系

脚本文件由一系列MATLAB指令组成，利用编辑器使其成为一个程序。以后，只要键入脚本文件名，就可以依次执行脚本中的指令，就像在MATLAB指令窗中按照同样的顺序键入这些指令一样。例如，要计算理想气体的体积随压力和温度变化的函数，我们先在编辑器中输入如下指令：

```
% ideal_gas.m-A sample script file
% Calculates the volume of an ideal gas
%
```

```
clear all; clc;  % clear all previous variables and command window
disp(' Calculating the volume of an ideal gas.')
R=8314;                 % Gas constant
T=input(' Vector of temperatures (K) =');
P=input(' Pressure (bar) =') * 1e5;
V=R * T/P;              % Ideal gas law
% Plotting the results
plot(T,V)
xlabel('T (K)')
ylabel('V (m^3/kmol)')
title('Ideal Gas Volume vs Temperature')
```

然后将这些指令保存到脚本文件 ideal_gas. m 中，再回到 MATLAB 指令窗，键入这个文件名：

```
>> ideal_gas
```

输入所需的数据，就可以得到计算结果。

初学者还可以利用实录指令 diary 创建脚本，方法如下。首先键入

```
>> diary mydiary
```

开始创建一个日志文件，然后在指令窗中一条条输入语句。比如，可以将以上脚本的语句输入，每一步都可以看到输出结果，需要的话，可以更正。输入完成并且确认结果正确之后，用如下指令关闭日志文件：

```
>> diary off
```

然后，可以编辑这个 mydiary 日志文件（日志文件不带扩展名），删除其中不需要的指令以及输出结果，将其保存为 M 文件，这样，就完成了一个脚本文件。

函数与脚本有所不同，函数接收某些输入数据，执行所需的各种计算，然后将计算结果返回给调用该函数的指令。用户可以开发自己的函数，并且像执行 MATLAB 其他函数一样调用这些函数。例如，将上述计算理想气体体积的脚本改写为如下函数，其通用性就更好，可以用于计算各种不同压力和温度下的气体体积。

```
function V=ideal_gas_func(T, P)
% This function calculates the specific volume of an ideal gas
```

```
R=8314;                      % Gas constant
for k=1:length(P)
      V(k,:)=R*T/P(k);   % Ideal gas law
end
```

如此例所示，函数的第一行为函数声明行，由关键词“function”开头，紧接着是输出参数、等号、函数名以及输入参数。紧跟函数声明行之后的注释段是函数的帮助信息，可以用如下 help 指令查阅：

```
>> help ideal_gas_func
```

此函数保存的文件名必须是 ideal_gas_func.m，然后就可以在工作空间、脚本文件、或者其他函数中调用此函数，例如：

```
>> P=[1:10]; T=[300:10:400];
>> vol=ideal_gas_func(T, P) ;
>> surf (T, P, vol)
>> view(135,45), colorbar
```

这几条指令执行的结果如图 A.17 所示。

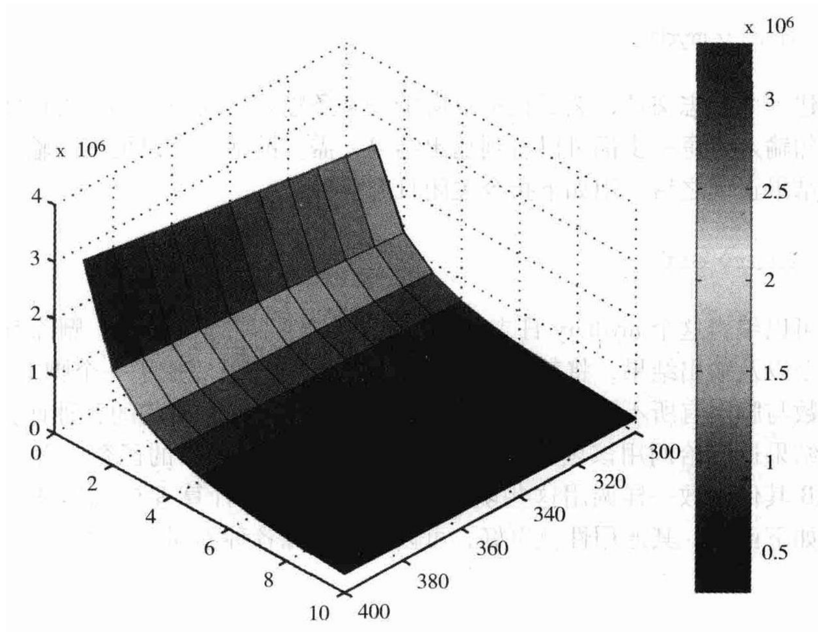


图 A.17 理想气体的体积随压力和温度变化的函数图

函数可以没有输入参数，也可以不返回任何计算结果。例如：上述脚本文件 ideal _ gas. m 也可以改写并保存为如下的函数形式：

```
function ideal _ gas
% Calculates the volume of an ideal gas
%
clear all; clc;  % clear all previous variables and command window
disp(' Calculating the volume of an ideal gas.')
R=8314;               % Gas constant
T=input(' Vector of temperatures (K) =');
P=input(' Pressure (bar) =') * 1e5;
V=R*T/P;              % Ideal gas law
% Plotting the results
plot (T, V)
xlabel('T (K) ')
ylabel('V (m^3/kmol) ')
title('Ideal Gas Volume vs Temperature')
```

这时，脚本和函数之间就没有什么区别了，通过文件名的调用，两者都可以在指令窗中运行，也可以被别的脚本调用。

A.7　程序流的控制

MATLAB 有数种程序流的控制结构，它们根据不同的数据值，控制程序按照不同的顺序执行语句。这些控制结构是 if、switch、for 和 while，简介如下。这里没有给出各段程序的运算结果，大家可以自己运行这些指令，查看其输出结果，程序的运行就作为练习留给读者。

"if . . . (else . . .) end" ——这个 if 控制结构使程序可以选择执行某些语句。下面这个例子中，只有当 x 的输入值大于或等于 0 时，$y = x$^2 这条语句才被执行：

```
x=input ('x=') ;
if x >=0
        y=x^2
end
```

如果增加 else 子句，那么，如果 if 语句的条件不满足，就会执行 else 子句中的语句:

```
x = input('x = ');
if x > =0
      y = x^2
else
      y = - x^2
end
```

“switch . . . case . . . end”——这是另一种条件控制形式，它根据变量的不同取值，分别执行不同的语句段。switch . . . case . . . end 结构比嵌套的 if 结构容易使用。例如:

```
a = input ('Give a value of a (1, 2, or 3) = ');
switch a
case 1
      disp ('One')
case 2
      disp ('Two')
case 3
      disp ('Three')
end
```

“for . . . end”——这个控制结构用于反复执行某个程序段，执行的次数为某个固定值，例如:

```
k = 0;
for x = 0:0.2:1
       k = k + 1
       y(k) = exp( - x)
end
```

“while . . . end”——这个控制结构用于反复执行某个程序段，直到某种情况出现为止，例如:

```
x = 0;
```

```
while x<1
      y=sin (x)
      x=x+0.1;
end
```

另外，还有 3 个程序流控制指令：break、pause 和 return。break 指令用于循环结构结束之前中止其执行。pause 指令会使程序暂停，等到键盘上有任意一键被按下后，再继续执行程序。例如：

```
k=0;
for x=0:0.2:1
     if k>3
        break
     end
     k=k+1
     y(k)=exp(-x)
     pause
end
```

"return" 指令则是结束函数的调用，返回到调用该函数的脚本或者指令行。

A.8　数据的显示、输入和输出

A.8.1　显示数据和结果

有多种不同的方法可以将数据等信息显示在监视器屏幕上、送到打印机打印、保存成文件、或者转换成其他格式输出。这里先介绍两个最基本的指令 disp 和 fprintf。disp 指令显示数组值，但不显示数组名，并且每次只能显示一个变量。例如：

```
>> X=[1 2 3; 4 5 6];
>> disp(X)
           1     2     3
           4     5     6
```

disp 指令的显示结果与表达式后不带分号的显示结果几乎相同，只是对于空数组，该指令无显示内容。disp 指令也可以用于显示文本，例如：

```
>> disp('This is the X array:'); disp(X)
This is the X array:
            1     2     3
            4     5     6
```

fprintf 指令可以将带格式的数据显示在监视器屏幕上或者写入文件中。例如，下面的指令用于显示文本和数据：

```
>> fprintf ('\n The value of position (2,2) of X = % g', X(2,2))
The value of position (2,2) of X = 5
```

专用的格式控制符 \ n、\ r、\ t、\ b 和 \ f 分别用于产生换行、回车、制表符、退格和换页。用“ \\ ”产生一个反斜杠，用“%%”产生一个百分号。

上面指令中的“%g”是一个控制数据显示格式的说明字符串，与 C 语言的格式转化说明相同，以字符%开头，之后可以有标志符、字段宽度和精度说明、附加格式说明符、以及格式字符 c、d、e、E、f、g、G、i、o、s、u、x、X 等。这些格式说明符的用法见表 A. 1。

下面是 fprintf 指令的几个应用示例，读者应该针对手头上的问题，设计合适的方式，简洁明了地显示数据和结果。

```
>> fprintf ('This is a test of the printing commands')
This is a test of the printing commands

>> a = -5.23 ;  b =3;  c =0.000000015;  d = 'Test';
>> fprintf ('Value of a = % 7.4f  b = % 2i  c = % 10g  d = % s \n',
a, b, c, d)
Value of a = -5.2300  b =3  c =1.5e -008  d = Test

>> fprintf ('% 12.2e', pi)
        3.14e +000

>> fprintf ('% 0.5e', pi)
3.14159e +000

>> fprintf ('% 0.5f', pi)
3.14159
```

```
>> fprintf ('% 10.5f', pi)
       3.14159

>> fprintf ('% 15.5g', eps)
       2.2204e-016

>> y=[1 2 3 4];
>> fprintf ('% 5.2f \n' ,y)
   1.00
   2.00
   3.00
   4.00
```

表 A.1　MATLAB 的输出格式说明符列表

格式说明符	输 出 形 式
%c	单个字符
%d	十进制带符号数
%e	用小写字母 e 表示的指数形式，例如 3.1415e+00
%E	用大写字母 E 表示的指数形式，例如 3.1415E+00
%f	定点数表示形式
%g	在%e 和%f 两种格式中选取较短的格式，不输出无意义的 0
%G	与%g 同，只是用大写字母 E
%i	十进制带符号数
%o	八进制无符号数
%s	字符串
%u	十进制无符号数
%x	用小写字母 a~f 表示的十六进制数
%X	用大写字母 A~F 表示的十六进制数

A.8.2　数据的存取

MATLAB 具有多种数据保存方法。这里，我们先把工作空间里已有的变量清除，再生成如下两个新变量：

```
>> clear
>> a=magic(3),  b=magic(4)  % magic is a special MATLAB ma-
trix
  a =
            8    1    6
```

```
          3     5     7
          4     9     2
b =
         16     2     3    13
          5    11    10     8
          9     7     6    12
          4    14    15     1
```

用如下 save 指令可以把 MATLAB 工作空间里的所有变量存入文件 f1. mat：

```
>> save f1
```

如果只要保存工作空间里的某些变量，那么在文件名之后列出变量名即可，例如：下面的指令只把变量 a 保存到文件 f2. mat 中：

```
>> save f2 a
```

以上生成的数据文件都带有扩展名“. mat”，只能由 MATLAB 读取。如果要用于别处，可以用如下方式把数据保存为文本格式：

```
>> save f3 b-ascii
```

这样，文件 f3 就是不带扩展名的文本文件。用 help save 指令可以查看 save 的所有不同用法。

用 load 指令可以把数据读入到 MATLAB 的工作空间中。如果文件是由 MATLAB 生成的并带有扩展名 . mat，则工作空间中出现的变量名与原来保存时的相同，例如：

```
>> clear
>> load f1
>> whos      % verifies the presence of these data
  Name        Size          Bytes  Class
  a           3x3              72  double array
  b           4x4             128  double array
Grand total is 25 elements using 200 bytes
```

但是，如果文件是文本文件，则工作空间中出现的变量名为文件名，如：

```
>> clear
>> load f3
>> whos
   Name       Size          Bytes    Class
   f3         4x4             128    double array
Grand total is 16 elements using 128 bytes
```

也可以用 MATLAB 编辑器生成数据文件，例如：先打开编辑器，输入矢量变量 $\boldsymbol{y}$ 的一列数值：

0.8345
0.0381
0.0163
0.0287
0.0220
0.0434

再把它存入文件 y.dat。

在 MATLAB 指令窗或者在脚本文件里，用如下 load 指令就可以读取这个文件中的数据：

```
load y.dat
```

这样，工作空间中就有了矢量变量 $\boldsymbol{y}$。

还有一种方法是用如下的 open 指令读取数据：

```
open y.dat
```

open 指令一执行就会自动打开如图 A.18 所示的输入向导（Import Wizard），用户可以在此指定数据的格式，然后再把数据读入工作空间，步骤如下。

按 Next 键，出现如图 A.19 所示的下一个窗，在此可以选择格式化多维数据的方式。

再按 Finish 键完成 y 数据的读取。Excel 等软件生成的电子表格数据，包括行和列的标题，都可以用这种方法读取。

还有一种在 MATLAB 指令窗或在脚本文件中读取数据的方法，就是先打开文件，再读数据，例如：

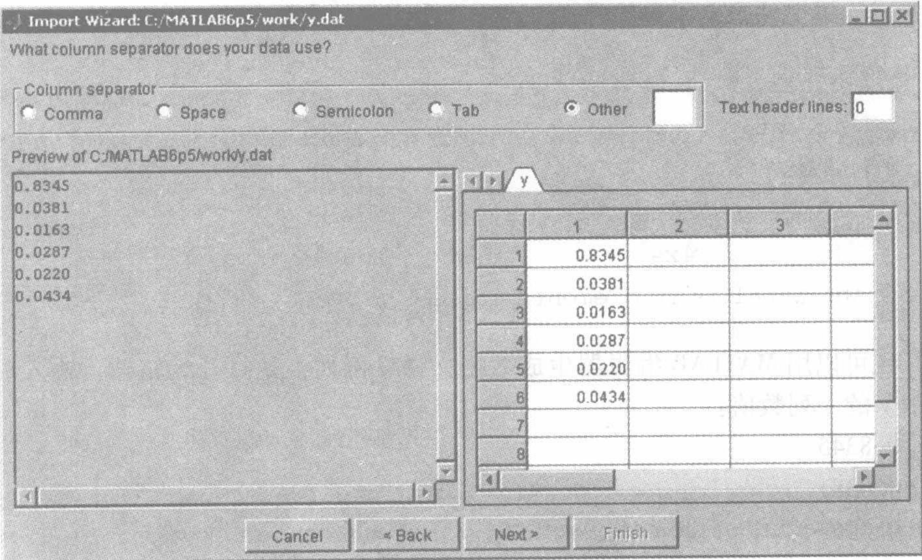


图 A.18 输入向导

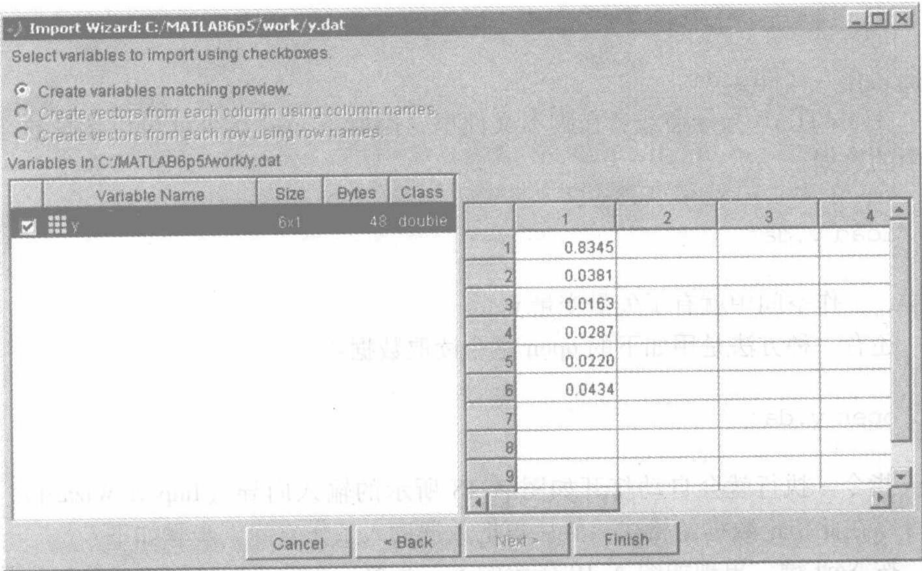


图 A.19 输入向导：格式化多维数据

```
fidy = fopen('y.dat');
%  fidy is the identification number for this file
y = fscanf (fidy, '% f')
y =
        0.8345
```

```
0.0381
0.0163
0.0287
0.0220
0.0434
```

读完数据之后，都要用 fclose 指令关闭文件：

```
fclose (fidy)
```

同样，也可以建立包含 $\boldsymbol{x}$ 和 $\boldsymbol{y}$ 两个变量的文件，并取名为 xy. dat：

```
0.0    0.8345
0.1    0.0381
0.2    0.0163
0.3    0.0287
0.4    0.0220
0.5    0.0434
```

其中，第一列是 $\boldsymbol{x}$，第二列是 $\boldsymbol{y}$，因此，数据的读取顺序必然是 $\boldsymbol{x}(1)$、$\boldsymbol{y}(1)$、$\boldsymbol{x}(2)$、$\boldsymbol{y}(2)$、$\boldsymbol{x}(3)$、$\boldsymbol{y}(3)$ 等等。可以用 MATLAB 编辑器编写如下脚本来读取 xy. dat 的数据：

```
% Opening and reading xy.dat file with two columns of data
%
clear
fidxy = fopen('xy.dat');
for i =1:6
      x(i) =fscanf(fidxy,'% f',1);  % reads one value of x at
a time
      y(i) =fscanf(fidxy,'% f',1);  % reads one value of y at
a time
end
fclose(fidxy);
x, y
```

其中的 fscanf 指令用于从文件中读取格式化数据。将这个脚本保存为 read _ xy. m 文件，并在指令窗中运行，则

```
>> read_xy
x =
    Columns 1 through 5
              0    0.1000    0.2000    0.3000    0.4000
    Column 6
      0.5000
y =
    Columns 1 through 5
      0.8345    0.0381    0.0163    0.0287    0.0220
    Column 6
      0.0434
```

请注意，读入的 ***x*** 和 ***y*** 是行矢量，而不是列矢量。

请读者也试一下用 open 指令和 load 指令读取该文件，即：

```
open xy.dat 或 load xy.dat
```

并观察读取结果的不同之处。

A.8.3 在程序中生成数据和保存数据

下面的程序首先生成一小段指数函数的数据表、再创建并打开一个名为 expon. dat 的文本文件，然后用 fprintf 指令将数据按照所需的格式写入该文件，最后再关闭文件。

```
x=0:0.1:1;
y=[x; exp(x)];
fid=fopen('expon.dat','w');
fprintf(fid, '% 6.2f % 12.8f\n',y);
fclose(fid);
```

下面用 load 指令读取这个文件，并把所有数据存放在一个矩阵变量中：

```
clear
load expon.dat
expon
expon =
          0    1.0000
```

```
0.1000    1.1052
0.2000    1.2214
0.3000    1.3499
0.4000    1.4918
0.5000    1.6487
0.6000    1.8221
0.7000    2.0138
0.8000    2.2255
0.9000    2.4596
1.0000    2.7183
```

根据需要，也可以用 A. 8. 2 节中所讲的方法打开 expon. dat 文件，将其读入后赋值给两个变量。

A. 9　符号运算

符号数学工具箱（Symbolic Math Toolbox）是 MATLAB 学生版的一个组成部分，它使 MATLAB 的数值环境有了解析计算，学生可以使用该工具箱进行符号数学计算和精确变量代数计算。读者最好通读一遍 MATLAB 所提供的有关符号数学示例（参见 Help/Demos/Toolboxes/Symbolic Math）。这里只列举其中的几条符号数学指令。

用如下 sym（ ）指令可以创建一个符号变量：

```
>> y = sym('y')
y =
y
>> om = sym('omega')
om =
omega
```

这些指令创建了一个显示为 y 的符号变量 y，以及一个显示为 omega 的符号变量 om。也可以用如下 syms 指令定义多个符号变量：

```
>> syms e f g
```

A.9.1 代数方程的符号求解法

这里，我们用符号函数 solve 来求解二次方程式

$$ax^2 + bx + c = 0$$

solve 函数的用法有以下几种：

1）假设未知量为 x，求函数等于 0 的解，则有

```
>> clear
>> x=solve('a*x^2+b*x+c')
x=
[ 1/2/a* (-b+(b^2-4*a*c)^(1/2)) ]
[ 1/2/a*(-b-(b^2-4*a*c)^(1/2)) ]
```

2）假设未知量为 x，由用户设定函数值等于 0，则有

```
>> x=solve('a*x^2 + b*x + c=0', 'x')
x=
[ 1/2/a*(-b+(b^2-4*a*c)^(1/2)) ]
[ 1/2/a*(-b- (b^2-4*a*c)^(1/2)) ]
```

3）将函数定义为符号 S，则有

```
>> syms x a b c S
S=a*x^2+b*x+c;
x=solve(S)
x=
[ 1/2/a* (-b+(b^2-4*a*c)^(1/2)) ]
[ 1/2/a* (-b- (b^2-4*a*c)^(1/2)) ]
```

4）假设未知量为 x，并且设定函数值等于 5，则有

```
>> x=solve('a*x^2+b*x+c=5','x')
x=
[ 1/2/a* (-b+(b^2-4*a*c+20*a)^(1/2)) ]
[ 1/2/a* (-b- (b^2-4*a*c+20*a)^(1/2)) ]
```

5）假设未知量为 a，而不是 x，则有

```
>> a=solve('a*x^2+b*x+c=5', 'a')
a =
- (b*x+c-5) /x^2
```

如果需要求解两个联立代数方程：

$$\alpha N_1 - \beta N_1 N_2 = 0$$
$$-\gamma N_2 + \delta N_1 N_2 = 0$$

则有

```
>> [N1,N2] =solve('alpha*N1 - beta*N1*N2=0', …
                  '-gamma*N2 + delta*N1*N2=0', 'N1,N2')
N1 =
[                 0]
[    1/delta*gamma]
N2 =
[                 0]
[     1/beta*alpha]
```

注意，命令行结尾处 3 个或多于 3 个的点“…”表示 MATLAB 语句延续到下一行。

A.9.2　微分方程的符号求解法

求如下微分方程的解：

$$\frac{dy}{dt} + 4y = e^{-t}$$

指令为

```
>> y=dsolve('Dy+4*y=exp(-t)')
y =
(1/3*exp (3*t) +C1) *exp (-4*t)
```

如果给定初始条件 $y(0) = 2$，则有

```
>> y=dsolve('Dy+4*y=exp(-t)', 'y(0)=2')
y =
(1/3*exp (3*t) +5/3) *exp (-4*t)
```

如果设定自变量的取值范围为 tt，求以上应变量 y 在此范围内的值，并作图（见图 A. 20），则指令为

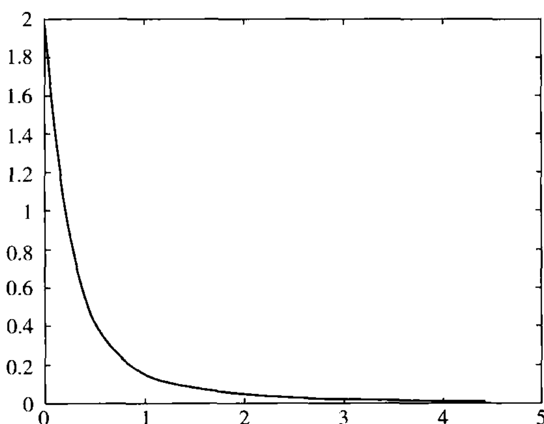


图 A. 20

```
>> tt=[0:.1:5];
>> for i=1:length(tt)
t=tt (i) ;
yy(i)=eval(y) ;
end
```

注意，符号解 y 中的 t 是一个标量，因此要用以上方法设置 t 的值。

```
>> plot (tt,yy)
```

如果需要求解两个联立微分方程：

$$\frac{dy_1}{dt}+4y_2=e^{-t} \text{ 初始条件为 } y_1(0)=1$$

$$\text{和}\frac{dy_2}{dt}=-y_1 \text{ 初始条件为 } y_2(0)=2$$

则有

```
>> [y1,y2]=dsolve('Dy1 + 4*y2=exp(-t) ', 'Dy2 = -y1', 'y1
(0)=1, y2(0)=2')
y1 =
```

```
-4/3*exp(2*t)+2*exp(-2*t)+1/3*exp(-t)
y2 =
2/3*exp(2*t)+exp(-2*t)+1/3*exp(-t)
```

请将求解结果代入微分方程进行验证。函数 diff 可用于求符号表达式的微分（见 A. 9. 3 节），例如

```
>> diff(y1) + 4*y2
ans =
exp(-t)
>> diff(y2)
ans =
4/3*exp(2*t)-2*exp(-2*t)-1/3*exp(-t)
```

如果设定自变量的取值范围，求应变量在此范围内的值，并作图（见图 A. 21），则为

```
>> t=[0:.01:1]; yy1=eval(y1); yy2=eval(y2);
>>plot(t,yy1,':', t, yy2)
```

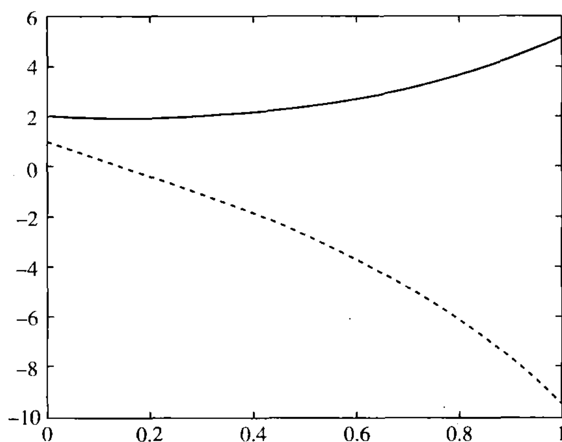


图　A. 21

A.9.3　符号微分

用如下指令清除前面的所有变量，并定义新的符号变量：

```
>> clear
```

```
>> syms x a f Re e
```

以下指令可以求各种函数的微分：

```
>> diff (x^2)
ans =
2 * x
>> diff (sin (x) )
ans =
cos (x)
>> diff (exp(x^2 )
ans =
2 * x * exp (x^2)
>> diff (log(x))
ans =
1/x
>> diff (acos(x))
ans =
-1/(1 - x^2) ^(1/2)
>> diff (acos(x))
ans =
-1/(1 - x^2) ^ (1/2)
>> diff (1/sqrt(f) + .86 * log(e/3.7 +2.51/Re/sqrt(f)) , f)
ans =
-1/2/f^ (3/2) - 10793/10000/Re/f^ (3/2) /(10/37 * e + 251/100/
Re/f^(1/2) )
```

A.9.4 符号积分

下面这个矢量中，前两个是对于 x 的积分，后两个是对于 a 的积分。

```
>> [ int(x^a), int(a^x), int(x^a, a) , int(a^x, a) ]
ans =
[ x^(a + 1)/(a + 1), 1/log(a) * a^x, 1/log(x) * x^a, a^(x + 1)/(x +
1)]
```

以下也是一个求积分的例子：

```
>> int('a*T+b*T^2+c*T^4', 'T')
ans =
1/2*a*T^2+1/3*b*T^3+1/5*c*T^5
```

还可以用如下方法设定积分的上下限：

```
>> int('a*T+b*T^2+c*T^4', 'T', 298,500)
ans =
80598*a+98536408/3*b+28899927176032/5*c
```

A.10　MATLAB 的工具箱

MATLAB 工具箱是一些特殊 M 文件的集合，专门用于某些特定领域的问题求解。工具箱可以任意选择，从 MathWorks 公司单独购买。表 A.2 列出了本书涉及到的一些工具箱。

MATLAB 学生版在基本 MATLAB 内容之外还加了 Simulink 仿真工具箱和符号数学工具箱。

表 A.2　MATLAB 的部分工具箱

工　具　箱	功　能
曲线拟合（Curve Fitting）	进行模型拟合和分析
图像处理（Image Processing）	进行图像处理、分析和算法开发
偏微分方程（Partial Differential Equation）	分析和求解偏微分方程
信号处理（Signal Processing）	进行信号处理、分析和算法开发
仿真（Simulink）	动态系统的建模、仿真和分析
统计（Statistics）	统计算法和概率模型的应用
符号数学（Symbolic Math）	进行符号数学计算和精确变量代数计算

A.11　参考文献

Hanselman, D., and Littlefield, B. 2005. *Mastering MATLAB 7.* Upper Saddle River, NJ: Prentice Hall.

Hahn, B. D. 2002. *Essential MATLAB for Scientists and Engineers.* Oxford, UK: Butterworth- Heinemann Publications.

Lyshevski, S. E. 2003. *Engineering and Scientific Computations Using MATLAB.* Hoboken, NJ: John Wiley & Sons, Inc..

附录 B　Simulink 简介

Simulink 是一个具有图形界面的 MATLAB 工具箱，用于系统建模和仿真，具有可视化编程语言和环境。Simulink 中，无论是内置函数还是用户自定义的函数都用模块表示，函数之间传输的数据则用连线表示。除了具有丰富的函数模块以外，它还有输入输出设备模块。Simulink 与 MATLAB 集成在一起，两者程序之间的数据转换很方便。本教程将结合书中所述的数值方法介绍 Simulink 建模与仿真的基本方法。UNIX、Macintosh 和 Windows 等环境都支持 Simulink，个人计算机的学生版 MATLAB 也包含有 Simulink。有关 Simulink 的详细信息请联系 MathWorks 公司。

B.1　动态系统仿真

如今，工程师们不需要实际的物理装置，只要使用各种随手可得的计算工具就可以进行动态系统仿真，也就是模拟。事实证明动态系统仿真对于系统设计的非常有用，如果不用仿真技术，而是建立物理系统模型，则既费时又费钱。Simulink 的主要优点是具有可视化编程环境，使用户能快速建模并即刻得到仿真结果。

信号流和逻辑流的概念

在 Simulink 中，模块（即函数）之间的数据流就是从一个模块的输出连接到下一个模块的输入的信号线，最终的输出信号可以送到示波器或显示器等接收模块显示出来，或者存入文件。数据可以从一个模块连接到另一个模块，也可以分支或者多路传输。仿真时，只是在离散的仿真时间（也称积分时间步长）上处理或转换数据。时间步长的选择取决于被仿真系统中变化最快的动态过程。

下面几节将介绍 Simulink 具有的各种不同模块。

B.2　启动 Simulink

在 MATLAB 指令窗中可以用两种方法启动 Simulink，如图 B.1 所示，一种是在指令提示符之后输入 Simulink 指令，另一种是按下工具栏上的 Simulink 按钮。

Simulink 启动之后出现的 Simulink 库浏览器是一个独立窗口，如图 B.2 所示，其中包含了标注为 A、B 和 C 的 3 个界面。A 界面是系统包含的 Simulink 模

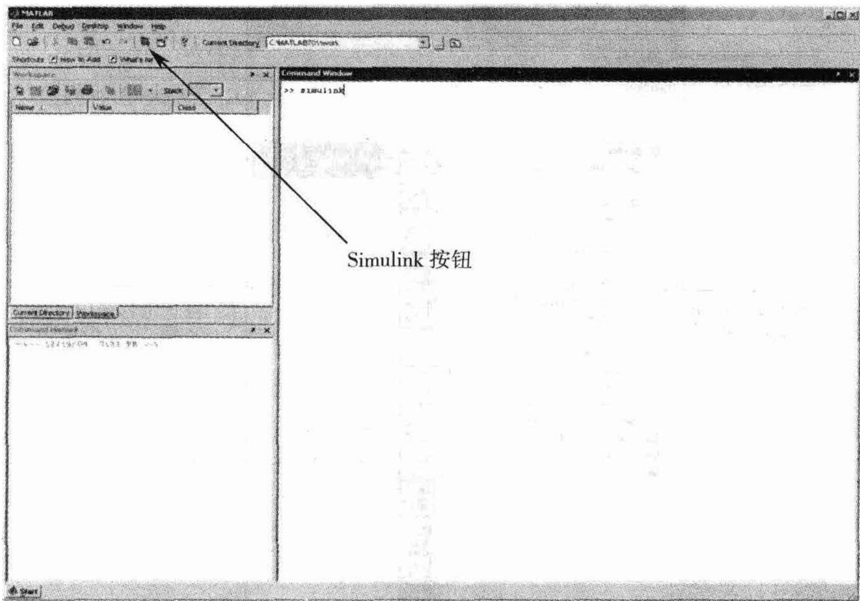


图 B.1　在 MATLAB 指令窗的指令提示符之后输入 Simulink 指令，或者单击图中箭头所指的工具栏上的浏览器按钮，即可以打开 Simulink 库浏览器（Simulink Library Browser）。

块库的目录列表，B 界面显示了每个模块库所包含的模块，C 界面则显示了所选中模块的定义，这个界面位于 A、B 两个界面的上方。

Simulink 模块的框图在模型窗中创建并修改，如图 B.2 所示，单击下 Simulink 库浏览器工具栏上的新建模型窗按钮，可以打开一个新的模型窗。

B.2.1　正弦波发生器的 Simulink 模型

按照如下步骤进行操作，可以在 Simulink 中建立一个正弦波发生器。

1. 将模块复制到模型中

图 B.3 显示了一个空白模型窗，只要把模块从库浏览器中拖到模型窗中就可以建立模型的框图。简单的演示模型用 3 个模块就能建立，比如要建立一个用于显示周期为 2π 的正弦波的模型，只需要从库浏览器中拖出以下 3 个模块（见图 B.4）：

1）信号源（Source）库中的正弦波模块（Sine Wave）；

2）接收器（Sinks）库中的示波器模块（Scope）；

3）数学运算（Math Operations）库中的增益模块（Gain）。

2. 将模块连接起来

为了显示周期为 2π 的正弦波波形，需要将正弦波模块的输出信号连接到增



图 B.2 Simulink 库浏览器

益模块的输入端，再将增益模块信号放大并输出到示波器模块，最后，由示波器模块在独立的窗口中显示出信号随时间变化的曲线。下面就来添加框图中正弦波模块到增益模块、以及增益模块到示波器模块的连接线（见图 B.5）。从信号起始点（模块的输出端）拖动鼠标到信号终止点（模块的输入端）就可以画出连接线。在画连接线时，要确保信号连到了所要到达的端口。当鼠标的光标接近某个模块的输出端时，Simulink 会将光标变为单十字线，按住鼠标左键拖出连线；当光标接近某个模块的输入端时，它又会变为双十字线，此时，放开鼠标即可。连接线添加之后，如果线上出现实心箭头，则表示信号连接正确；如果出现空心箭头，则信号还没有与两个模块都连上。要修正这种开路信号线，只需把空心箭头作为模块的输出端，继续用上述方法将连线拖拉到下一个模块的输入端即可。图 B.5 所示是已完成的模型框图，图 B.6 显示的是示波器的输出信号（参见下面的运行仿真）。

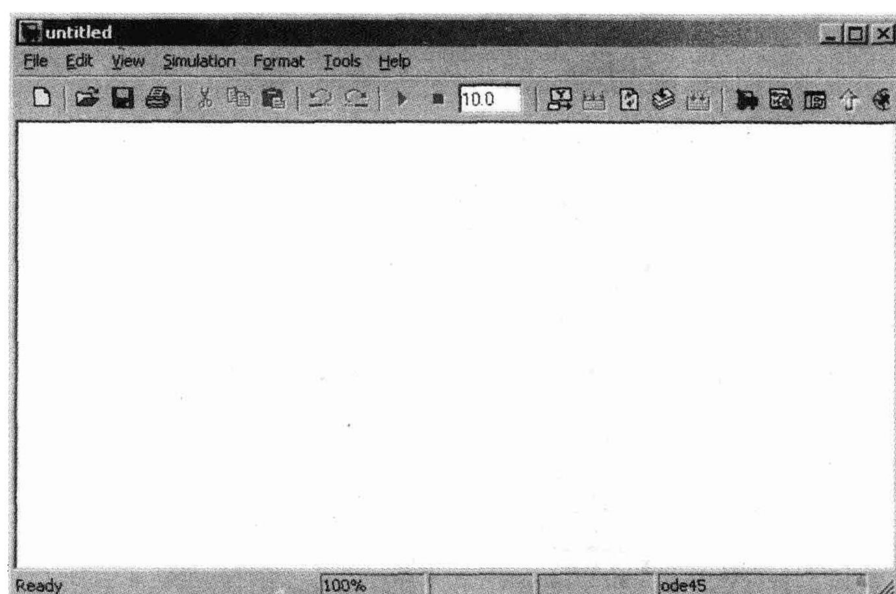


图 B.3　空白 Simulink 模型窗

表 B.1 总结了建立 Simulink 模型的有关编辑操作。

表 B.1　Simulink 的编辑操作

操　　作	击键以及鼠标动作
从模块库中复制一个模块	用鼠标左键直接将模块拖拉到模型窗，或者利用编辑（EDIT）菜单的复制（COPY）和粘贴（PASTE）操作来完成
在模型中复制模块	按住 Ctrl 键，并用鼠标左键选中模块，然后拖拉到新的位置
显示模块参数	双击模块
翻转模块	Ctrl-F
旋转模块（每次顺时针旋转 90°）	Ctrl-R
修改模块名	单击模块的标签，并将光标置于所需位置
断开模块的连接	按住 Shift 键，并将模块拖拉到新的位置
画一条斜线	按住 Shift 键，同时按住鼠标左键拖动
线的折弯	将光标移动到线上需要折弯的位置，按住 Shift 键，同时按住鼠标左键拖动连接线

3. 运行仿真

拉下模型窗的仿真菜单并单击启动选项，或者如菜单中所提示的，直接按 Ctrl-T 键，就可以启动模型的仿真计算。本例是一个很简单的模型，仿真瞬间就

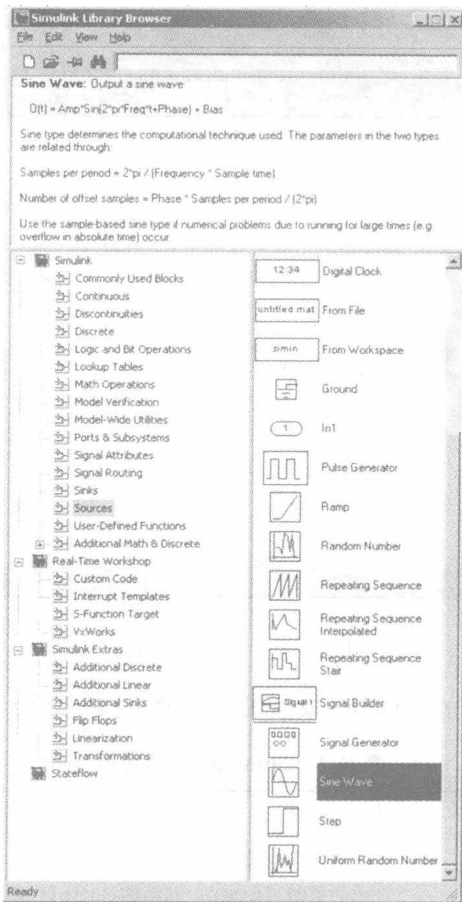


图 B.4 创建正弦波发生器的 Simulink 模型（图中所示模块被拖拉到模型窗）

结束。如果系统比较复杂，则可以从模型窗下方的运行时间上看到仿真的进度。仿真结束之后，双击示波器可以查看增益模块的输出结果，本例的输出是一个幅值随时间变化的正弦曲线。示波器窗口出现后，如果需要根据窗口的尺寸调节所显示图形的大小，则可以单击其工具栏上形似双筒望远镜的坐标刻度调节按钮。图 B.6 所示就是显示输出结果的独立窗口。

B.2.2 仿真模型的修改

1. 正确连接信号线

在画信号连接线时，不必担心走过的途经是否合适，这些连线会进行自动调整。等到模块都连接好之后，可以重新调整模块所处的位置，使系统框图简洁明了。只要将模块选中并拖到合适的位置即可，信号线依旧连接，并且会自动调整

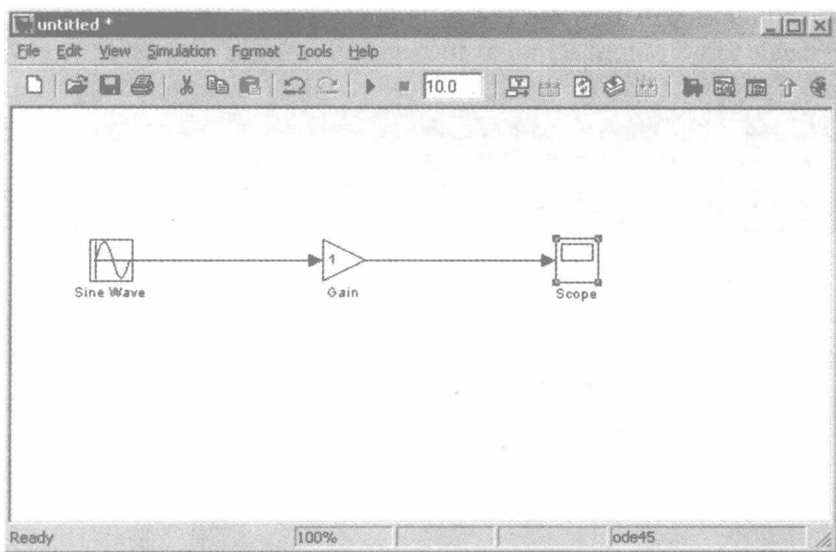


图 B.5　Simulink 模块的连接（将模块的输出拖拉到信号传递途径中的下一个模块的输入端）

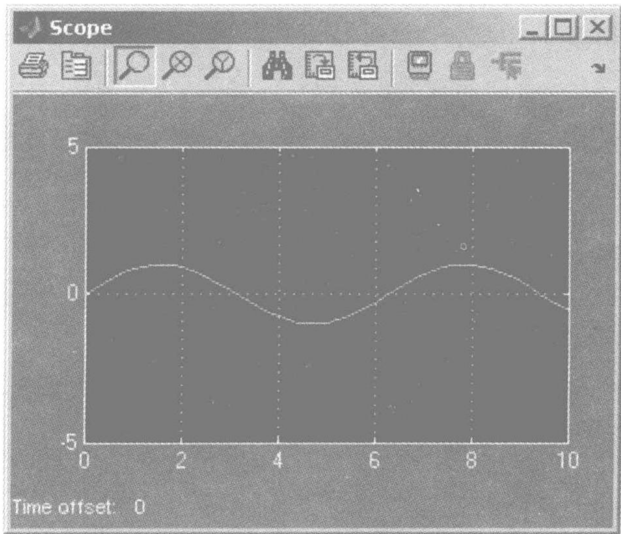


图 B.6　图 B.5 所示模型的示波器输出结果

到合适的路径。

有些模型中需要将信号线分支，分别连接到两个或更多个不同的输入端。进行这个操作时，先把鼠标的光标移动到需要分支的点上，再用 Ctrl 键加鼠标左键或者只按住鼠标右键，拖出新的连线，连接到目标点即可。图 B.7 显示了用这

种方法分支正弦波输出信号线。这里，已将两个示波器模块重新命名为“Scope 1”和“Scope 2”。

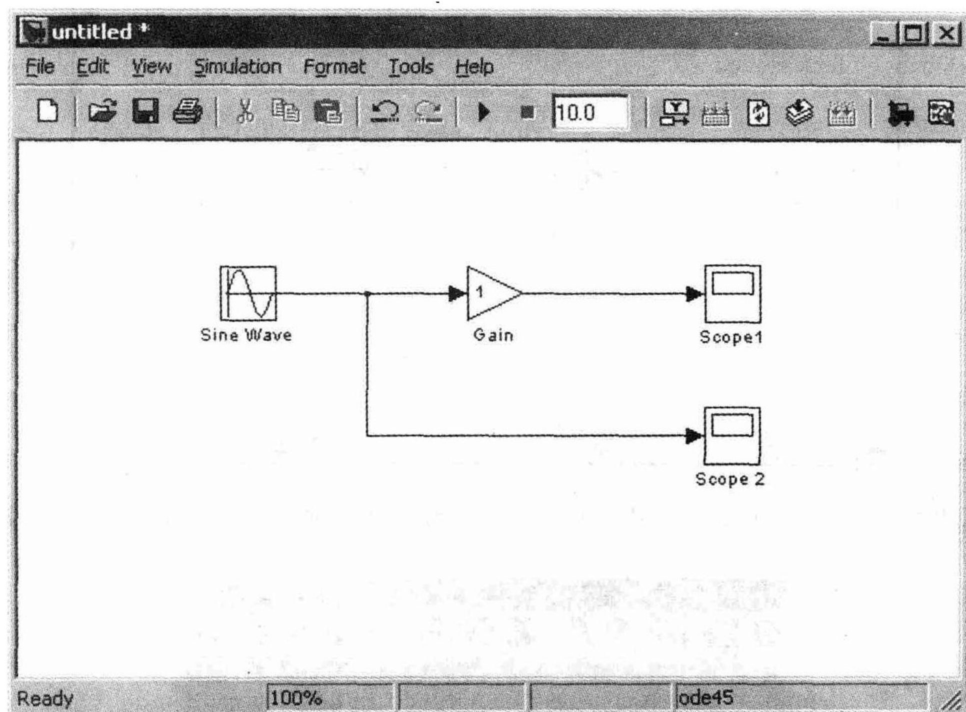


图 B.7 信号线的分支：在连线上按住鼠标右键并拉出，或者按住 Ctrl 键同时用鼠标左键拉出

2. 修改模块：输入模块的参数

双击模块可以显示模块的参数。例如，要改变上述增益模块的增益值，如图 B.8 所示，双击此增益模块之后，将增益值改为 2。

3. 仿真参数的配置

（1）仿真时间

启动并运行图 B.8 所示系统的仿真有 3 种方法：从仿真菜单中选择“开始”选项（Start）；单击模型窗工具栏上的“开始”/“暂停”按钮（Start/Pause）；按 Ctrl-T 键。其仿真输出如图 B.9 所示。注意，因为经过了增益模块的放大，第一个示波器（Scope 1）显示的正弦波幅值为 2。而第二个示波器（Scope 2）直接显示信号发生器的输出，因此其信号幅值为 1。

注意，在模型窗 Start/Pause 按钮的右侧以及示波器的 x 轴上都可以看出，此处的仿真时间长度为 10s。通过改变模型窗工具栏上的仿真时间，或者修改仿真菜单中的参数配置项，都可以改变仿真时间的长度。如图 B.10 所示为 20s 仿真

的示波器输出结果。

（2）状态变量及其算法

就生物医学系统模型而言，即使不是所有模型都是如此，也有很多模型具有这样的结构，就是其输出是变量自身先前输出值的函数，而输出变量往往是时间函数。在 Simulink 等仿真工具中，这种变量被称为状态变量。计算时，在每个时间步长都要保存计算结果，以便用于计算下一个输出值。

用 MATLAB 编程时，要先把状态变量数据保存在 MATLAB 工作空间的变量中，然后再用于计算后面的输出值。而在 Simulink 中，如果有模块需要用其自身先前的输出值来计算当前的输出值，那么就意味着需要保存状态，Simulink 就会自动保存模型所包含的状态变量。

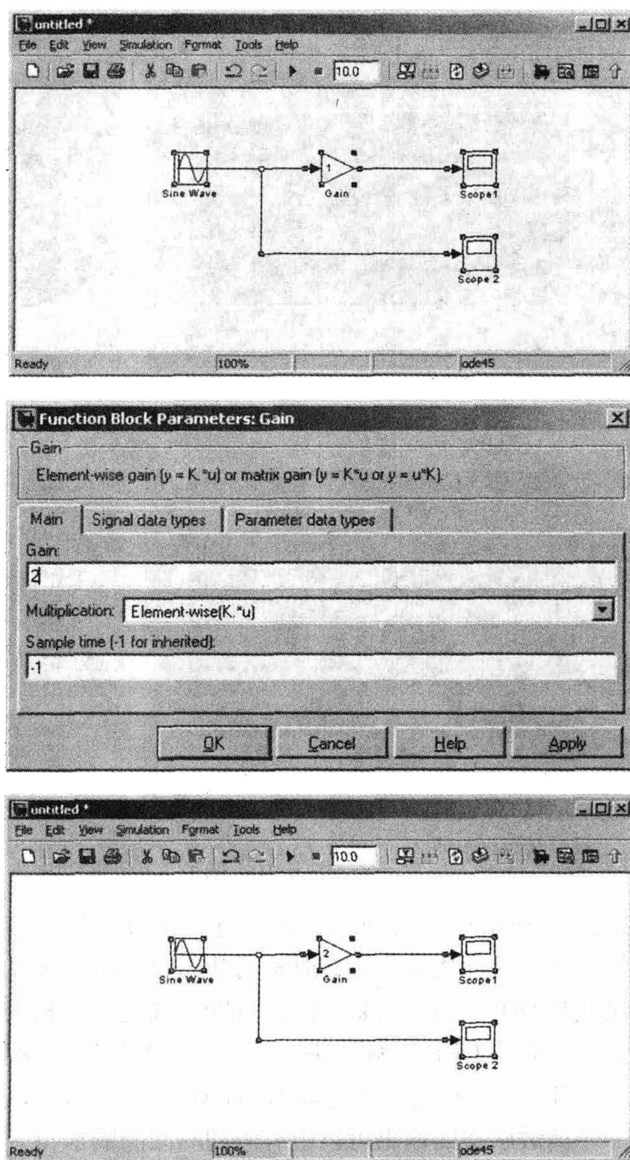


图 B.8　改变增益模块的参数

如果选用数值方法实现模型的仿真计算，也就是求解常微分方程（Ordinary Differential Equation，ODE），那么，就必须了解状态以及状态求解器的概念。Simulink 模型可以有两种不同的状态：连续状态和离散状态。如果模型含有连续状态，则可以用第 7 章介绍的那些 ODE 算法；如果模型只含有离散状态，则其仿真可以用离散算法实现。

如果 Simulink 模型中含有一个连续状态，那就意味着其中有某个计算需要先进行微分，然后再用这个微分做积分。积分器的输出要用于计算下一个微分值。

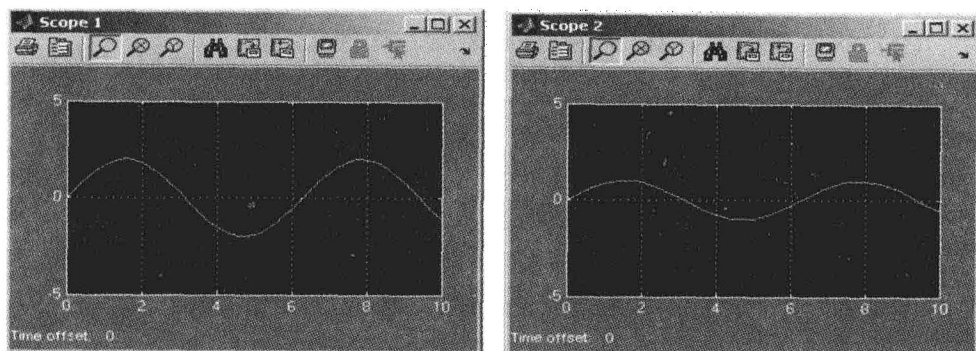


图 B.9　图 B.8 模型的 10s 仿真结果

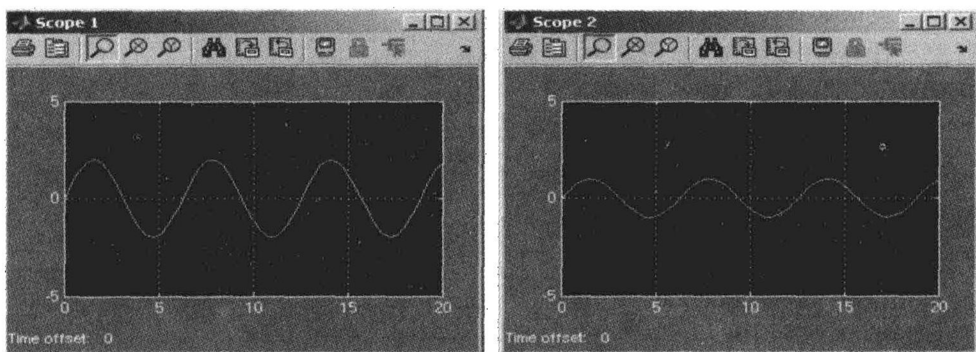


图 B.10　图 B.8 模型的 20s 仿真结果

这样，Simulink 模型就要用到“连续模块库”中的积分模块，以及一组计算微分的模块。这种建模方法有时称为状态空间法。构成积分模块输入信号的那一组模块就是 ODE。Simulink 的仿真精确度取决于步长的大小以及所采用的 ODE 求解算法。有关 ODE 求解算法以及它们的 MATLAB 函数实现请参考第 7 章的详细内容，Simulink 直接使用这些 MATLAB 函数来求解包含连续状态的模型。

离散状态的输出可以由有限的时间间隔求得。Simulink 有两种离散状态求解算法：即固定步长算法和变步长算法。前者按照固定的周期更新状态的值，后者则不定期地更新状态的值，或者采用能保证所有离散状态都得到更新的最小步长。当离散步长的大小接近于 0 时，则离散状态法近似于连续状态法。

在更改仿真时间长度的同一个菜单里，可以选择仿真中采用的 ODE 算法。用于连续状态系统的各种 ODE 算法参见表 7.2。记住，Simulink 还有两种离散状态系统的求解器算法。

在开始仿真之前，Simulink 会先检查模型，确定其是否含有连续状态或者离

散状态。如果模型包含有积分器模块、状态空间模块、或者传递函数模块，就要用连续状态，如果使用离散算法，就会出错。图 B. 8 的模型没有积分器，因此可以用离散算法。图 B. 11 显示了使用变步长离散算法求解图 B. 8 模型的结果。

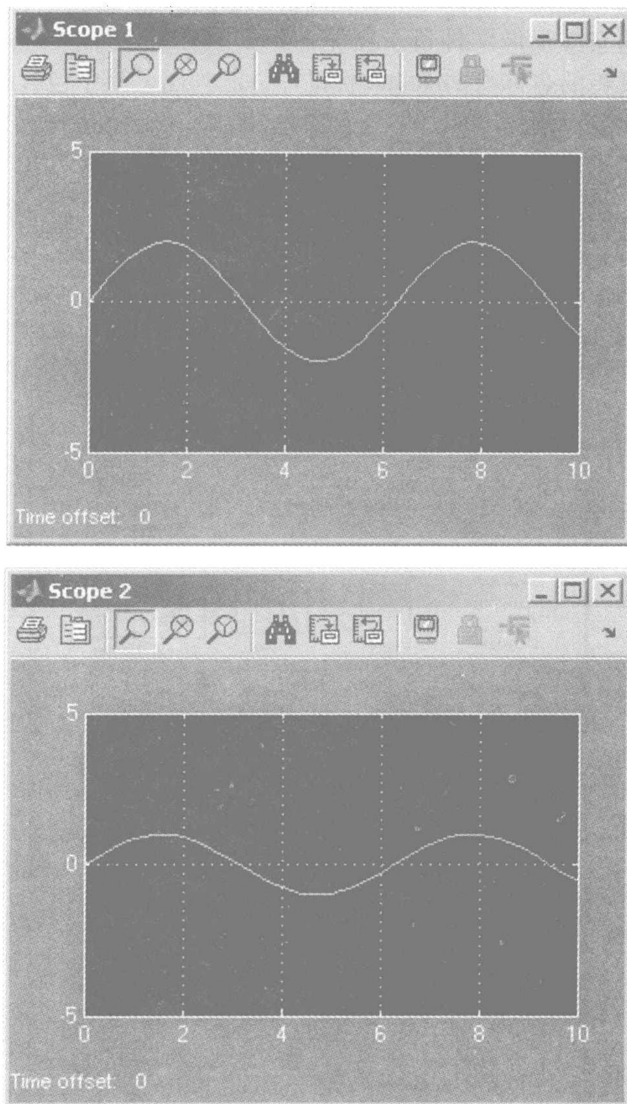


图 B. 11　图 B. 8 模型的仿真结果（采用了离散算法）

果然，由于模型中没有连续状态，仿真结果没什么不同。

Simulink 的仿真参数还有很多，比如数据的输入输出和优化等，这些内容就不在此讲述了，读者可以参考 Simulink 的帮助信息。

B.3 Simulink 模块库

打开 Simulink 库浏览器，左下方界面显示的就是 Simulink 具有的模块库目录列表，此界面的安排很像文件目录浏览器。如果在安装 Simulink 时有附加工具箱，则附加工具箱的名称与 Simulink 标准库一起列出，其中包含的模块库则以子目录的形式列在工具箱的下面。子目录中包含的模块显示在右下方的界面中。下面几张图分别显示了几个主要的模块库。

图 B. 12 所示是信号源模块库，其中包括的数据源和信号源（即信号发生器）作为模型或子系统的输入。信号发生器有常量、周期函数、噪声和时间标记等多种不同的形式。

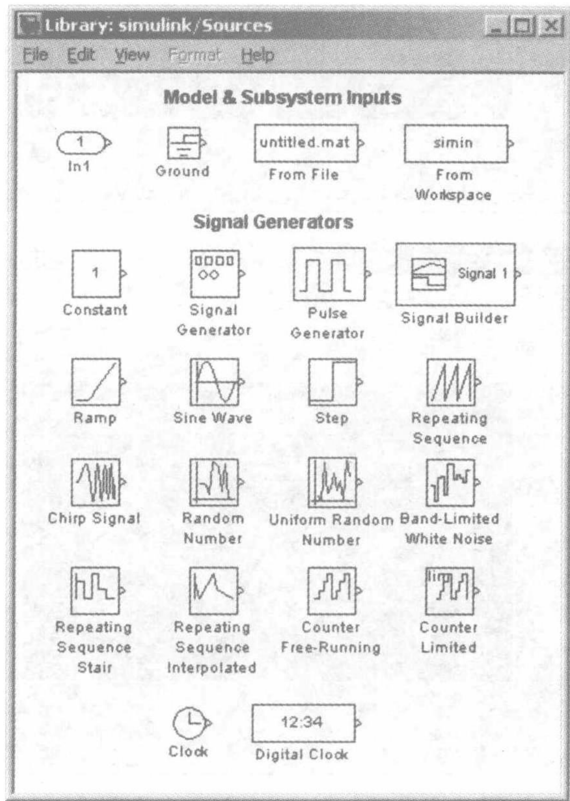


图 B. 12 Simulink 信号源模块库中的模块

图 B. 13 所示是接收器模块库，其中的模块用于接收信号或者终止仿真。可分为 3 个子库：模型或子系统的输出端口模块（用于将信号传送给其他模型）、

数据浏览器模块（包括示波器等）和仿真控制模块（用于终止仿真）。

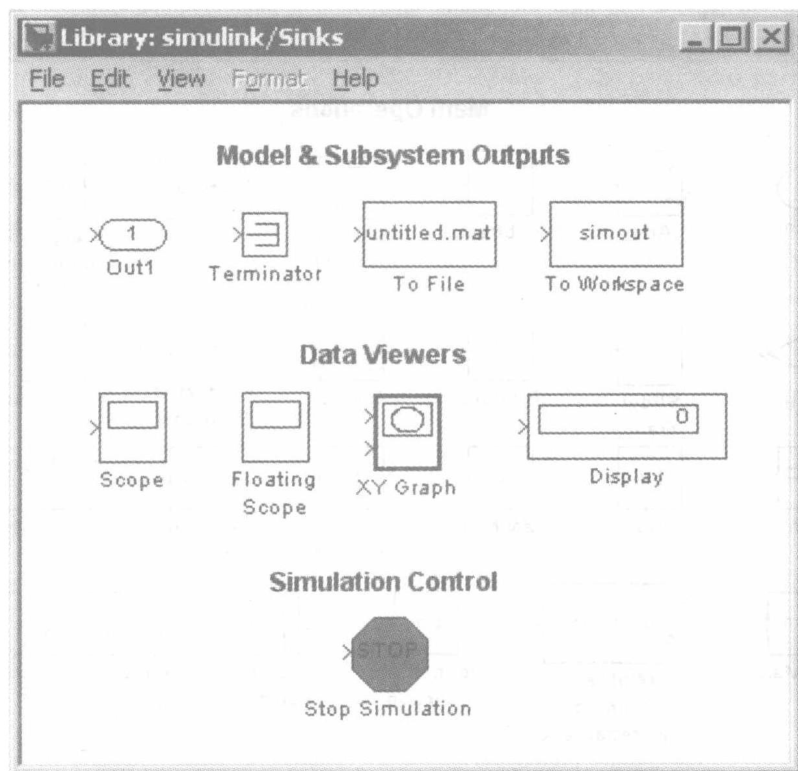


图 B. 13　Simulink 接收器模块库中的模块

我们已经用过的还有一种模块是如图 B. 14 所示的数学运算模块。这个模块库有一大堆函数，包括标量和矩阵的代数运算、复数的各种转换运算等。

要了解不同 ODE 算法产生的影响，就需要在模型系统中引入连续状态和离散状态的概念。图 B. 15 就是连续模块库包含的模块，有积分器和微分器。任何包含一个或多个这种模块的模型都要用连续状态 ODE 求解算法（见表 7. 2）。

通用模块库随时都会用到，它含有许多模块，比较复杂的模型也可以用这个库中的模块来建立。图 B. 16 列出了其中的一些常用模块。

以上这些模块库包括了生物医学系统模型仿真所需要的主要模块，有些比较复杂的模型可能还需要用到其他模块库，例如：

1）端口和子系统（Port & Subsystem）：该模块库中的模块用于定义和连接多个模型。就像 MATLAB 程序需要分为多个脚本和函数一样，较大的 Simulink 模型应该分成多个子系统，这样便于处理。端口和子系统模块库中的模块就是用于定义子系统并且完成模型之间的信号传递。

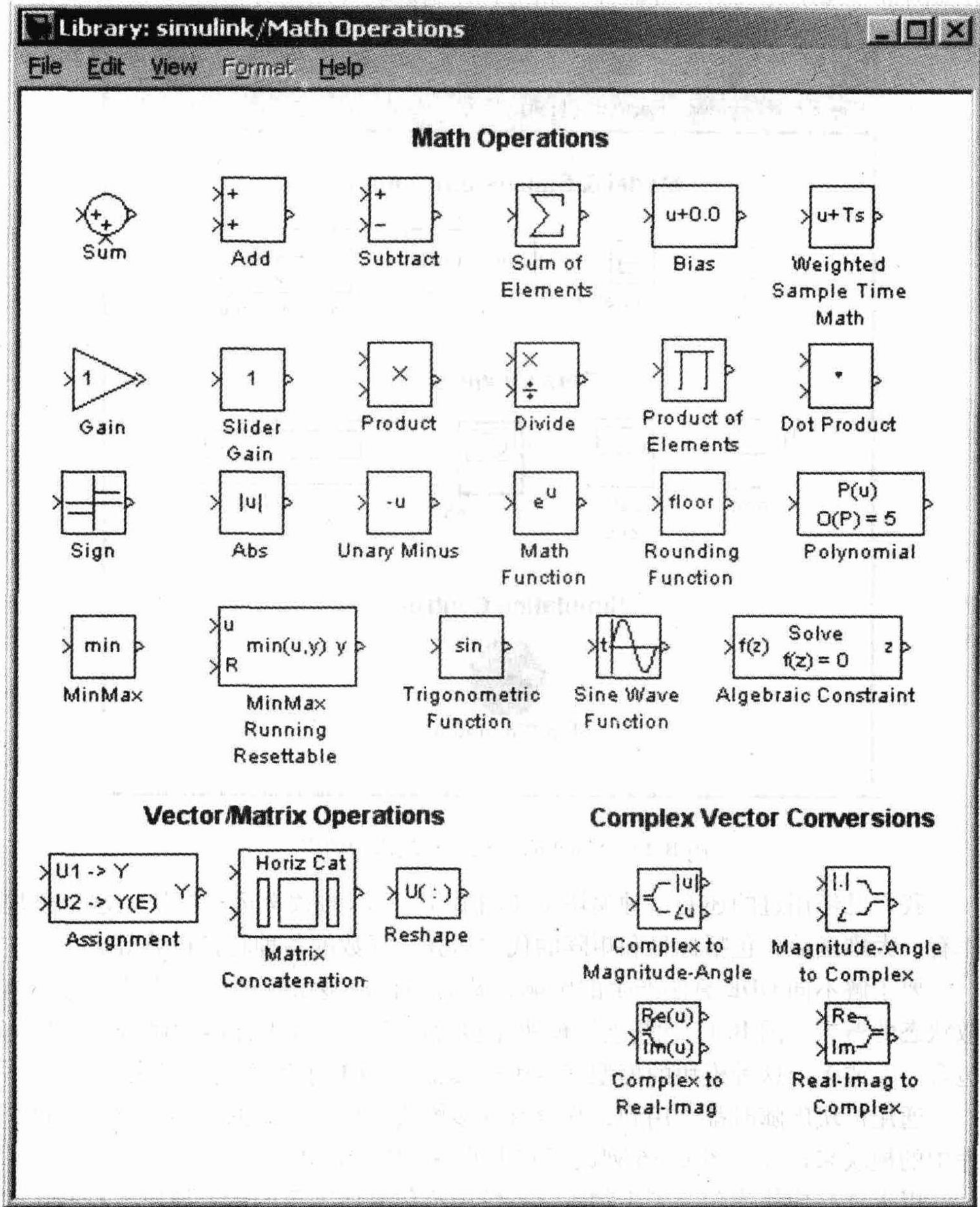


图 B. 14　Simulink 数学运算模块库中的模块

2）信号路由（Signal Routing）：该模块库用于处理模型中的信号流，其中多数常用模块是配对使用的。

① 总线（Buses）模块用于减少模型中的连线数量。

② 多路切换（Multiplexers）模块，也就是 Mux/DeMux 对。用于将多个信号组合成为一个信号，或者反之，从一个合成信号中分出多路信号。

③ 传出、传入（Goto/From）模块，用于无实线连接的模块之间的信号传送，常用于两个模型之间的信号传递。

3）用户自定义函数（User-Defined Functions）：用于创建实现自定义运算的模块，用户定义的运算可以是 MATLAB 程序、C 语言程序或者用 C 语言语法表示的代数表达式等。

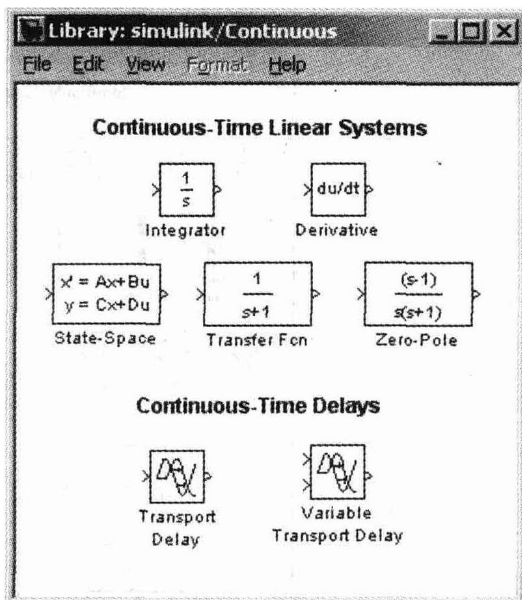


图 B.15 Simulink 连续模块库中的模块

Simulink 标准工具箱中的其他模块库在此就不再给出图示了，标准 Simulink 的帮助信息中有详细资料，这些模块库是：

1）非连续（Discontinuities）模块库，用于处理模型中的饱和和死区等非线性关系。

2）离散（Discrete）模块库，利用 Z 变换方法（一种到离散信号空间的变换）对传递函数进行滤波和模拟等计算。

3）逻辑运算和位运算（Logic and Bit Operations）模块库，用于单个二进制位或字的逻辑运算。

4）查表（Lookup Tables）模块库，可以完成某个离散数据域到另一个离散域的函数映射。

5）模型验证（Model Verification）模块库，用于建立自检模型，这些模型可以根据需要随意关闭或启用。

6）模型实用（Model-Wide Utilities）模块库，用于注释模型。

7）信号属性（Signal Attributes）模块库，用于处理模型中的数据类型。

8）附加数学和离散化（Additional Math & Discrete）模块库，用于处理信号幅值的表示，包括溢出、浮点数的定点表示等。

除了标准工具箱之外，还有一个 Simulink 附加工具箱，其中最重要的是附加接收器（Additional Sinks）模块库，含有频谱分析模块等；以及转换（Transformations）模块库，用于实现坐标系统或者计量单位的变换等。

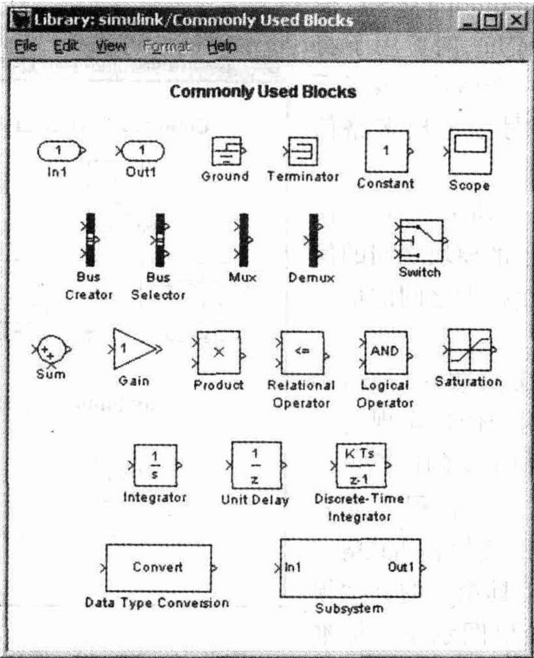


图 B. 16 Simulink 通用模块库中的模块

B.4 构建模型

前面 B. 2 节中所介绍的初步例子只是演示一下 Simulink 的操作方法，并不是 Simulink 应用于生物医学系统模拟的完整教程。基于已介绍的示例和模块，本节讲述几个构建 Simulink 模型的实例。

B.4.1 代数运算、信号路由以及 MATLAB 变量

B. 2 节的示例中有一个正弦波发生器和两个示波器，其中示波器 2 显示正弦波发生器的输出，而示波器 1 则显示正弦波发生器输出被放大 1 倍之后的信号。

假设现在增加一个正弦波发生器，并且两个正弦波的幅值都设为 3，频率都为 0. 5Hz，但是第二个正弦波相对于第一个正弦波有 0. 25rad/s（弧度/秒）的相位差。要求计算两个信号之差以及差的积分，再将计算结果与两个正弦波信号一起显示在示波器上，并且把两个信号之差和差的积分存入 MATLAB 文件用于以后的计算。仿真的时间长度设定为 15s。则其解析式为

$$\int[\sin(0.5t)-\sin(0.5t+0.25)]\mathrm{d}t$$

此例中要引入总线和积分器这两个新模块，模型中的多路信号可以通过总线模块一起传送。总线把多路信号组合在一起，并将信号传输给可以接收多路输入的模块，本例的示波器就是这种模块。积分器则输出其输入信号的积分值。双击积分器可以查看其参数，其中最重要的是初始值，其默认值为 0。下面是创建本例模型的步骤（括号内标注的是所添加模块的模块库名称）：

1）创建两个正弦波发生器（信号源模块库），并双击模块名，分别将两个正弦波发生器改名为“Sine Wave 1”和“Sine Wave 2”。

2）添加一个减法模块（数学运算模块库），将其两个输入端分别连接到两个正弦波发生器的输出端。

3）添加一个积分器（连续模块库），并将其输入端连接到减法模块的输出端。

4）添加第二个示波器，并同第一步所示，改变示波器标签名称，以区别两个示波器。

5）添加两个总线创建器（信号路由模块库），它们的输出总线分别连接到两个示波器的输入端。连接到第一个示波器的总线，其输入信号包括正弦波 1、正弦波 2 以及两个正弦波之差；连接到第二个示波器的总线，其输入则包括正弦波 1、正弦波 2 以及两个正弦波之差的积分。

6）添加两个“工作空间”模块（接收器模块库），用于将计算结果传送给 MATLAB，以便进行后续计算。第一个“工作空间”的输入端连接两个正弦波之差，第二个则连接两个正弦波之差的积分。在这两个模块的模块参数中可以设置 MATLAB 工作空间变量的变量名。

7）添加第三个示波器，显示正弦波 1 的输出信号，作为参考。

完成所有这些模块的添加之后，按照以上步骤中所指定的方式把各个输入端和输出端连接起来，整个模型如图 B. 17 所示。图 B. 18 显示了 3 个示波器的输出结果。该模型计算了略有相位差的两个正弦波信号之差，将结果显示在 Simulink 示波器上并输出到 MATLAB 工作空间的变量中。

上述解析表达式的积分为

$$-2\cos(0.5t) + 2\cos(0.5t + 0.25)$$

仿真计算运行之后，MATLAB 工作空间会出现两个变量，3 个示波器上则显示大约 1¼周期的各个输出信号。在实际计算机监视屏上，示波器 1 和示波器 2 的各路输出曲线采用了不同的颜色：黄色是总线的第一路输入信号，粉色是第二路，绿色则是第三路。

正如预计的那样，与两个信号本身的幅值相比，两个正弦波之差很小（见图 B. 18 示波器 1 的第三路信号）。增加信号的相位差，则两个信号之差的幅值会相应增大，读者不妨试一下。利用输出到 MATLAB 变量的数据可以计算信号

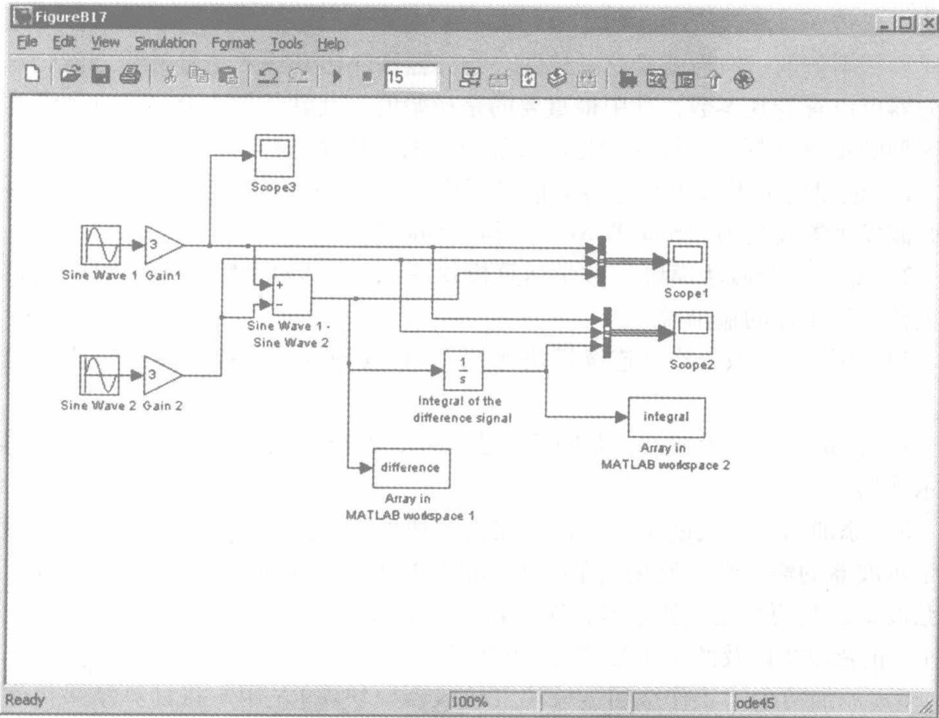


图 B. 17 该模型演示积分器、总线以及 MATLAB 输出的用法

的精确幅值和统计量，例如，执行 MATLAB 指令

```
max (difference)
max (integral)
```

可以得到如下结果：

```
>> max (difference)
ans =
    0.7467
>> max (integral)
ans =
    1.6824
```

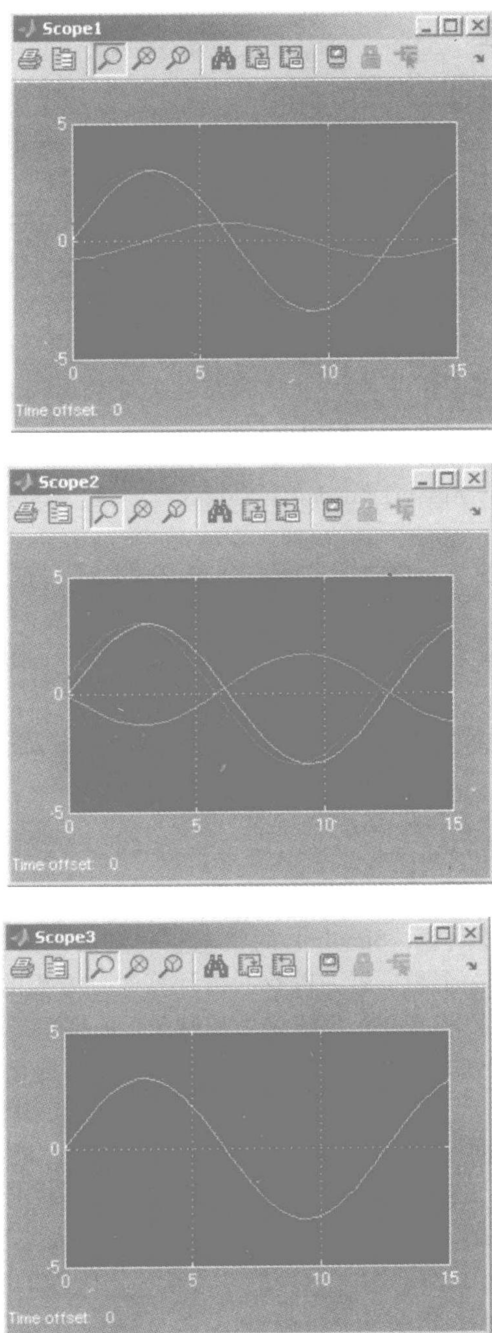
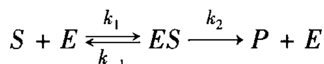


图 B.18　图 B.17 所示模型的 3 个示波器的输出信号

B.4.2 微分方程组

在第 7 章中，例 7.2 的微分方程组是用四阶龙格-库塔法（即 ode45 求解器）求解的。这里，再用这个例子来演示一下如何用 Simulink 仿真求解常微分方程组。

例 7.2 模拟某种酶 E 的酶促反应。酶 E 催化底物 S，通过形成中间复合物 ES，使底物 S 转化为产物 P。其化学反应式如下：



$$S + E \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} ES \xrightarrow{k_2} P + E$$

题目要求读者计算酶促反应将 99.9% 底物转化为产物所需要的时间。该化学反应的数学模型由如下 4 个微分方程组成：

$$\frac{\mathrm{d}[S]}{\mathrm{d}t} = -k_1[S][E] + k_{-1}[ES] \qquad [S]_0 = 1.0$$

$$\frac{\mathrm{d}[E]}{\mathrm{d}t} = -k_1[S][E] + k_{-1}[ES] + k_2[ES] \qquad [E]_0 = 0.1$$

$$\frac{\mathrm{d}[ES]}{\mathrm{d}t} = k_1[S][E] - k_{-1}[ES] - k_2[ES] \qquad [ES]_0 = 0$$

$$\frac{\mathrm{d}[P]}{\mathrm{d}t} = k_2[ES] \qquad [P]_0 = 0$$

其中速率常数为

$$k_1 = 0.1(\mu\mathrm{M})^{-1}\mathrm{s}^{-1} \qquad k_{-1} = 0.1\mathrm{s}^{-1} \qquad k_2 = 0.3\mathrm{s}^{-1}$$

下面的 Simulink 仿真基于一种被称为状态空间解法的微分方程求解方法。对于单个初始值的 ODE 问题，Simulink 的状态空间解法包括 3 部分：一个示波器用于显示输出信号，一个积分器，以及一组计算微分方程的模块（见图 B.19）。双击积分器模块，可以将其初始值设定为初始条件所提供的值。

本题目有 4 个微分方程，这 4 个方程都要建立各自的状态空间解法模型，其中 4 个“子系统”的计算结果经过线性组合，传送给各个积分器。在第 7 章的 MATLAB 解法中，用两张图分别显示了底物与产物以及酶与复合物两对变量；这里的 Simulink 模型与例 7.2 相似，用两个示波器显示输出结果，信号由总线组合之后送给各个示波器。图 B.20 所示就是 Simulink 的完整模型。

模型右下方部分用于计算 99.9% 底物完成转化所需的时间，它由两个模块组成：一是把［S］输出值与 0.001 进行比较，这是一个常数比较模块（逻辑和位运算模块库），一旦限定条件达到，仿真计算就结束。另一个就是仿真终止模块（接收器模块库），它由比较模块的布尔输出值控制。在仿真计算过程中，化学反应的时间值被存入名为 time999 的 MATLAB 工作空间的变量中。仿真结束

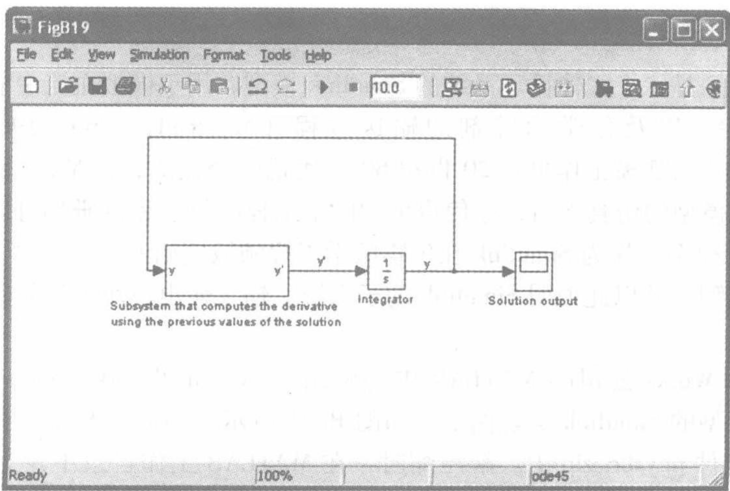


图 B.19　ODE 状态空间求解的一般结构（初始条件作为积分器模块的初始值）

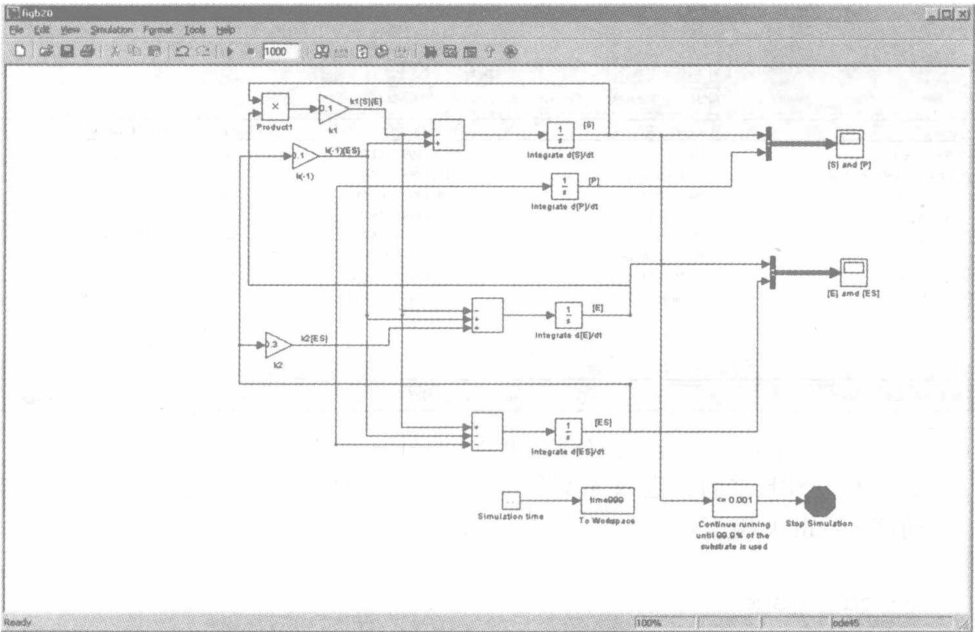


图 B.20　例 7.2 求解的 Simulink 模型

时，该数组变量的最后一个值就是 99.9% 底物完成转化的时间。仿真结果与例 7.2 相同。

B.4.3 PHYSBE 及其子系统

PHYSBE 是一个人体循环系统的 Simulink 模型，用于模拟血流中氧气、营养物质、热量、以及化学示踪剂的输送过程（McCleod，1966，1968）。尽管 PHYSBE 的主要建模工作早在 20 世纪 60 年代就已经完成，但是，后来用 Simulink 实现了模型的仿真之后，才使得生物医学工程的教学和科研都可以比较方便地使用这个模型。作为 Simulink 在生物医学工程领域应用的一个实例，下面介绍 PHYSBE 模型，并以此说明 Simulink 的子系统、输入输出模块等信号路由工具的使用方法。

从 MathWorks 公司的 MATLAB 中心网站（www. mathworks. com）可以下载 PHYSBE 模型的 Simulink 实现例子，如图 B. 21 所示，一共有 8 个文件，一起保存在压缩文件 physbc. zip 中。解压缩时，在 MATLAB 工作目录中建立 physbe 子目录，无论用哪种压缩工具（WinZip、7-zip、Windows XP 的资源管理器、或者是 Mac OS X 的 ZipIt 等），使用给定的路径名就很容易操作。

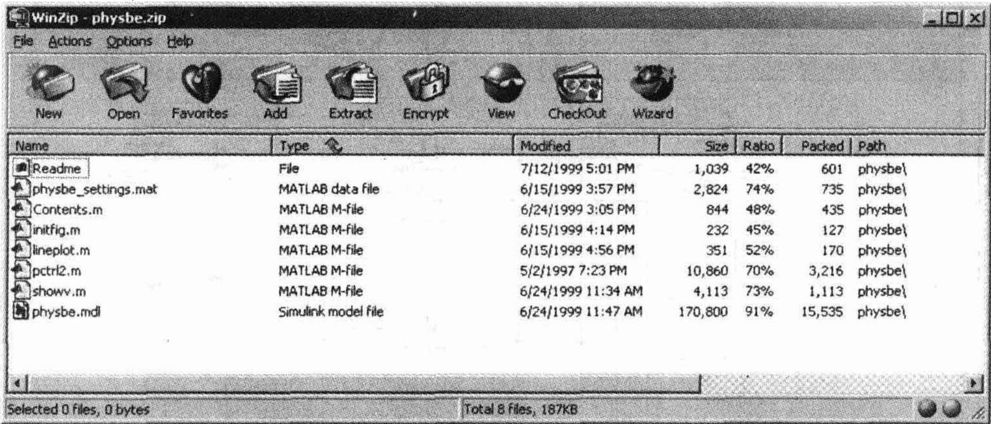


图 B. 21 循环系统仿真器 PHYSBE 的 . m 文件和 . mdl 文件

安装 PHYSBE 之后，进入 MATLAB，在 MATLAB 指令提示符下键入如下指令启动该 Simulink 模型：

```
>> cd physbe
>> physbe
```

这时，会出现 5 个窗口：PHYSBE 模型窗、PHYSBE 控制中心、监视心脏血压和心脏血容量的两个示波器，以及一个选取式示波器。图 B. 22 所示是最高级别的 PHYSBE 循环系统模型，由 9 个子系统组成，另加一个分析子系统（右下方）。双击子系统方框，可以将其打开。例如，双击标有 LUNGS 的子系统方框

可以查看肺模型，该模型的详细内容如图 B. 23 所示。

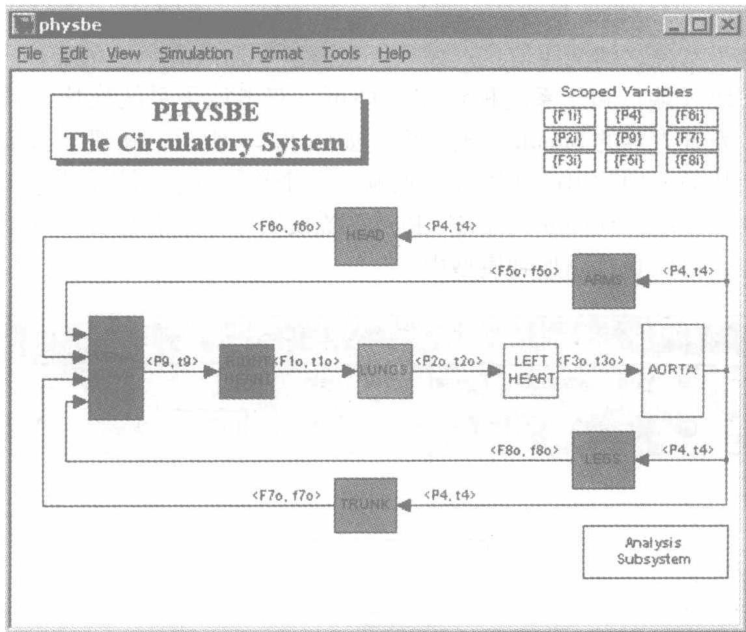


图 B. 22　PHYSBE 的 Simulink 模型（由图中所示的 10 个子系统组成）

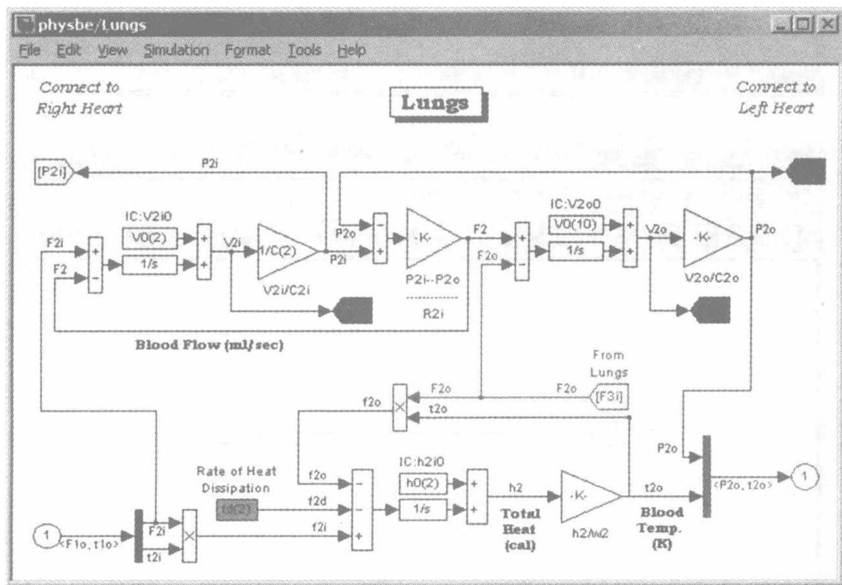


图 B. 23　PHYSBE 的肺模型（是整个模型中的一个子系统，其输入为 P2i 和 < F1o, t1o >；输出为 V2i、V2o、P2o 以及 < P2o, t2o >）

图 B. 23 的 Simulink 肺模型是子系统的一个例子。该子系统本身就是一个 Simulink 模型，具有输入输出等信号以及各种函数，可以在该子系统内以及整个系统的其他子系统之间交换信息。如果要在模型中加入一个子系统，只要添加一个子系统模块（端口和子系统模块库）即可，双击该子系统模块，可以新开一个显示该子系统内容的 Simulink 模型窗。如图 B. 24 所示，新建子系统的初始结构只有一个从输入到输出的连接线。先删除这个连接线，再添加所需要的模块，就可以建立子系统。Simulink 的端口和子系统模块库中还有包含详细结构的子系统实例模块，可以直接添加到模型中。

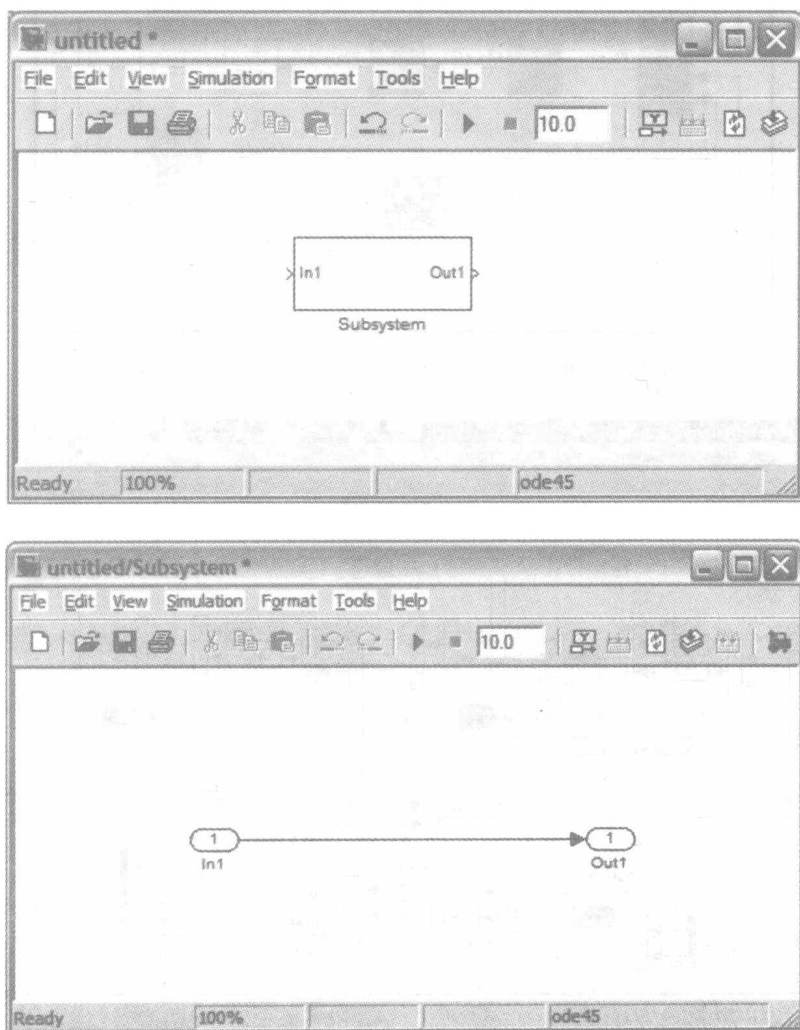


图 B. 24　新建的子系统模型：（上图）打开之后只有输入到输出的一条连接线（下图）

McCleod 的原始论文（McCleod，1966，1968）以及 MathWorks 公司的 PHYSBE 网站上都有关于 PHYSBE 模型参数的解释。各子系统之间的信号传输线用一个或者多个参数名标出，信号线标签用尖括号表示，如：<Bn（i/o），Hn（i/o）>，其中第一个参数 B 是血流参数，第二个参数 H 是热流参数。尖括号用于表示这些参数是由总线传输的，总线把两个信号组合成一个输出，传输到下一个模块的输入端时信号又被分开，这种信号的输入输出用图 B. 24 所示的子系统的输入、输出模块实现。

例如：图 B. 22 中的“RIGHT HEART”的输出是包含输出流速和输出温度两个信号<F1o，t1o>的总线，这两个信号在肺模型 LUNGS 中被分开，分别命名为 F2i 和 t2i，也就是进入 LUNGS 模型的血流速度和热流温度。信号接收方的数字是指 LUNGS 在 PHYSBE 控制中心所处的行（见图 B. 25），即 F2i 指进入 LUNGS 的流速。PHYSBE 控制中心是一个用 GUIDE 设计和开发的图形化用户界面。GUIDE 作为用户界面编辑器是 MATLAB 的一个组成部分，是创建复杂模型用户界面的功能强大的工具，其使用方法本书不做介绍。

PHYSBE Control Center

File Edit View Insert Tools Desktop Window Help

Save **PHYSBE Control Center** Reset

	Ri	Ro	C	W	A	V0	h0	fe	fd
R.Heart	0.0128	0.0111	75	600	0	150	0	0	0
Lung	0.1429	0	7.519	1000	0	120	0	0	0
L.Heart	0.0588	0.0125	80	600	0	150	0	0	0
Aorta	0	0	1.25	0	0	100	0	0	0
Arms	5.15	10	4.25	7000	3670	280	0	0	0
Head	2.58	5	1.21	4500	1400	80	0	0	0
Trunk	0.67	1.42	34	53000	6000	2250	0	0	0
Legs	2.58	5	11.1	18500	7000	730	0	0	0
V.Cava	0	0	250	0	0	500	0	0	0
Lung2*	0	0	30.3	0	0	240	0	0	0

图 B. 25 PHYSBE 的控制中心（可以在执行仿真之前修改和设置其中的参数）

每个 PHYSBE 子系统都是自左向右处理信号流。例如：图 B. 23 的 LUNGS 子系统中，标有 1 的圆圈所表示的输入是包含 F1o 和 t1o 两个模型参数的总线。输出则在右边，用了一个包含 P2o 和 t2o 信号的总线。另外，还用了输入输出模块（From、Goto）在子系统之间传送其他信号，例如，肺的血压 P2i 用输出模块

送给了右心模型，即在 LUNGS 子系统中用了一个 Goto 模块，在 RIGHT HEART 子系统中用了一个 From 模块。

与其他 Simulink 模型一样，通过仿真下拉菜单或者在键盘上键入 Ctrl-T，可以运行 PHYSBE 模型，如图 B. 26 所示为使用预定参数运行仿真之后，在标记为心脏血压（Heart Pressure）和心脏容量（Heart Volume）两个示波器中输出的结果。

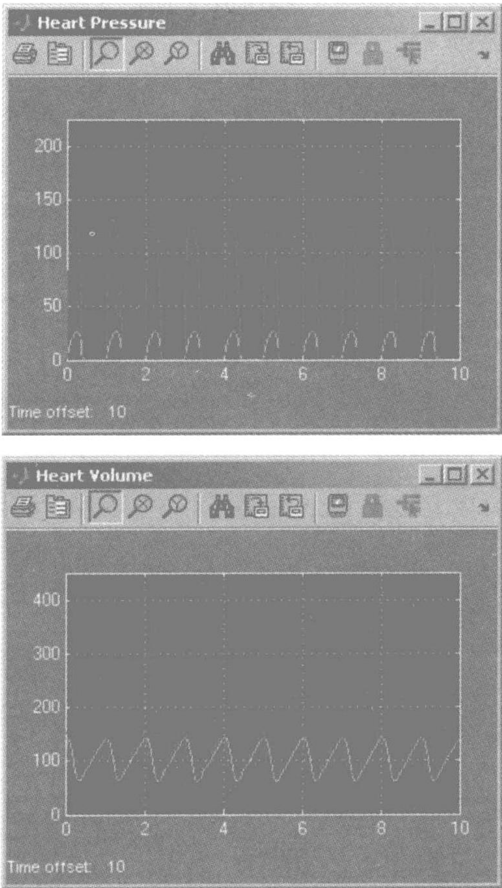


图 B. 26　显示在 Heart Pressure（上图）和 Heart Volume（下图）两个示波器中的 PHYSBE 输出结果(仿真时没有改动任何预定参数)

在 PHYSBE 模型的分析子系统中可以看到这两个示波器的输入，Heart Pressure 示波器的输入是 P1 和 P3，Heart Volume 示波器的输入是 V1 和 V3，4 个信号分别来自于 RIGHT HEART 和 LEFT HEART 两个子系统。

PHYSBE 还有一个选取式示波器，可以显示多路信号，但与 Heart Pressure 和 Heart Volume 两个示波器不同，它不使用总线。选取式示波器的用法是：同时

打开示波器和信号选择器，选中要显示信号左边的小方块，就可以在示波器中显示该信号。图 B.27 显示了这两个窗口，信号选择器有两个界面，分别列出了各个子系统以及其中包含的信号。

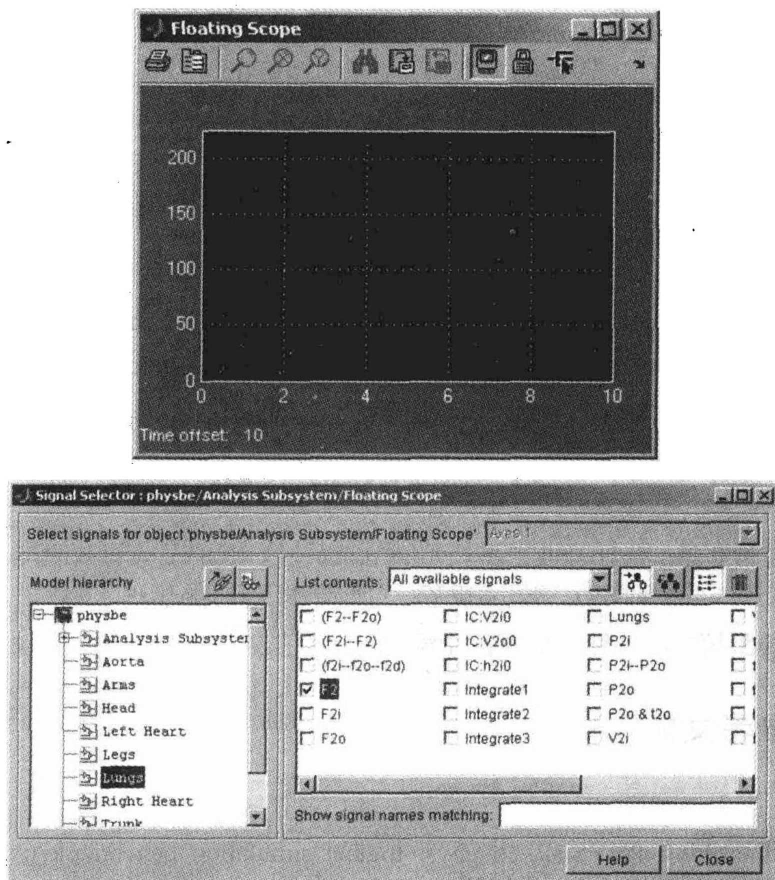


图 B.27 PHYSBE 的选取式示波器（上图）及其信号选择器（下图）

例如，假设要在同一个选取式示波器中显示头部（HEAD）血压 P6、右心脏（RIGHT HEART）血压 P1 以及腿部（LEGS）血压 P8，则打开选取式示波器及其信号选择器，依次选定这 3 个子系统中的这些血压信号，然后关闭信号选择器，运行仿真，就可以得到图 B.28 所示的选取式示波器的输出结果。

信号选择器的左边是子系统模型列表，右边是被选中子系统包含的信号，其中的 F2 信号已被选中。

可见，躯干血压较高，并且它比头部和腿部的血压要平稳。

本附录的目的是介绍 Simulink 模型的各个组成部分，阐明仿真心血管循环系统之类复杂问题的基本要素。读者可以阅读第 10 章的实例，了解如何应用

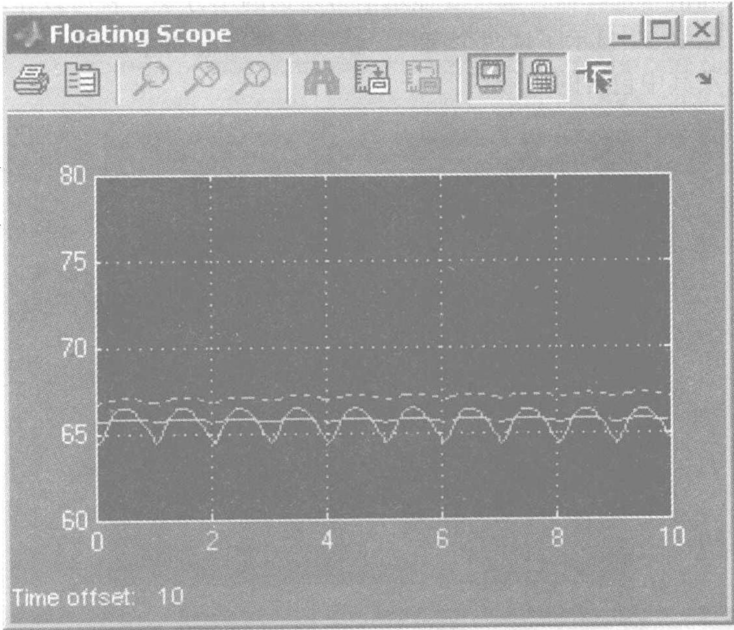


图 B.28 在 PHYSBE 选取式示波器上显示 3 个血压信号：头部血压 P6（正弦样曲线）、躯干血压 P7[㊀]（虚线）以及腿部血压 P8（实线）

PHYSBE 模型模拟心血管系统的各种病理。

B.5 参考文献

McLeod J, (1966). PHYSBE...a physiological simulation benchmark experiment. *SIMULATION*, **7(6)**:324-329.

McLeod J, (1968). PHYSBE...a year later. *SIMULATION*, **10(1)**:37-45.

㊀ 原文这里的 P7 与正文中所述 P1 不相符合。——译者注

附录 C　线性代数及其相关的 MATLAB 指令

C.1　矩阵和矢量的运算

下面这个矩阵 $\boldsymbol{A}$ 是一个元素按照行和列排列的（$m \times n$）数组：

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

在 MATLAB 中，行元素用逗号分开，行与行之间则用分号或换行符分开，即

```
>> B=[1, 3,5;2,0,4; 1, 5, 6]
B =
      1     3     5
      2     0     4
      1     5     6
```

矩阵的迹（trace）是方阵主对角线上元素之和。MATLAB 的求迹指令是

```
>> trace (B)
ans =7
```

具有 n 个元素的单列矩阵被称为 n 维列矢量，如：

```
>> x=[1; 3; 5; 2; 0]
x =
    1
    3
    5
    2
    0
```

具有 n 个元素的单行矩阵则被称为 n 维行矢量，如：

```
>> y=[5, 6, 7, 8, 9]
y=
     5     6     7     8     9
```

具有相同行数和列数的矩阵可以进行矩阵的加减运算。

如果矩阵 $\boldsymbol{A}$ 的列数与矩阵 $\boldsymbol{B}$ 的行数相等，则 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 之间可以进行矩阵的乘法运算。如果这个条件满足，称矩阵 $\boldsymbol{A}$ 和矩阵 $\boldsymbol{B}$ 是一致的。（$m\times n$）的矩阵 $\boldsymbol{A}$ 与（$n\times p$）的矩阵 $\boldsymbol{B}$ 相乘，结果是具有 m 行和 p 列的（$m\times p$）阶矩阵，如：

$$\boldsymbol{AB}=\underset{(4\times 3)}{\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{bmatrix}}\underset{(3\times 2)}{\begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix}}$$

$$=\underset{(4\times 2)}{\begin{bmatrix} (a_{11}b_{11}+a_{12}b_{21}+a_{13}b_{31}) & (a_{11}b_{12}+a_{12}b_{22}+a_{13}b_{32}) \\ (a_{21}b_{11}+a_{22}b_{21}+a_{23}b_{31}) & (a_{21}b_{12}+a_{22}b_{22}+a_{23}b_{32}) \\ (a_{31}b_{11}+a_{32}b_{21}+a_{33}b_{31}) & (a_{31}b_{12}+a_{32}b_{22}+a_{33}b_{32}) \\ (a_{41}b_{11}+a_{42}b_{21}+a_{43}b_{31}) & (a_{41}b_{12}+a_{42}b_{22}+a_{43}b_{32}) \end{bmatrix}}$$

矩阵的乘法运算是不可交换的，也就是 $\boldsymbol{AB}\neq\boldsymbol{BA}$。要注意，两个矩阵对应因素之间的相乘用“. *”运算符号表示，即

$$\boldsymbol{A}.*\boldsymbol{B}=\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}.*\begin{bmatrix} b_{11} & b_{12} \\ b_{21} & {}_{22} \end{bmatrix}=\begin{bmatrix} a_{11}b_{11} & a_{12}b_{12} \\ a_{21}b_{21} & a_{22}b_{22} \end{bmatrix}$$

矩阵 $\boldsymbol{A}$ 的转置 $\boldsymbol{A}^{\mathrm{T}}$ 定义为 $\boldsymbol{A}$ 的行与列的交换，在 MATLAB 中表示为 $\boldsymbol{A}'$。如果 $\boldsymbol{A}=\boldsymbol{A}^{\mathrm{T}}$，则 $\boldsymbol{A}$ 是对称矩阵。

对角线上的元素均为 1，其余元素均为 0 的矩阵称为单位矩阵，单位矩阵总是对称方阵。例如，（3×3）阶单位矩阵为

$$\boldsymbol{I}=\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

MATLAB 函数 eye（n）返回的就是一个（$n\times n$）阶单位矩阵。如：

```
>> eye (3)
ans =
     1     0     0
```

```
      0    1    0
      0    0    1
```

矩阵 $\boldsymbol{A}$ 的逆矩阵 $\boldsymbol{A}^{-1}$，是与矩阵 $\boldsymbol{A}$ 相乘以后可以得到单位矩阵的那个矩阵，即

$$\boldsymbol{A}\boldsymbol{A}^{-1} = \boldsymbol{I}$$

如果矩阵 $\boldsymbol{A}$ 是方阵，并且是非奇异的，也就是矩阵行列式的值不为0，则 $\boldsymbol{A}$ 的逆矩阵存在。用以下 MATLAB 指令可以方便地计算矩阵行列式的值，检验矩阵 $\boldsymbol{B}$ 是否是非奇异的：

```
>> B=[1,3,5;2,0,4; 1,5,6]
B =
      1    3    5
      2    0    4
      1    5    6
>> det (B)
ans =
      6
```

同样，用 MATLAB 指令也很容易计算矩阵的逆，上面这个矩阵 $\boldsymbol{B}$ 的逆为

```
>> inv (B)
ans =
    -3.3333     1.1667     2.0000
    -1.3333     0.1667     1.0000
     1.6667    -0.3333    -1.0000
```

两个矩阵相除，即 $\boldsymbol{A}$ 除以 $\boldsymbol{B}$，等于矩阵 $\boldsymbol{A}$ 与矩阵 $\boldsymbol{B}$ 之逆的乘积，即 $\boldsymbol{A}/\boldsymbol{B} = \boldsymbol{A}\boldsymbol{B}^{-1}$。而左除 $\boldsymbol{A} \setminus \boldsymbol{B}$ 则等于 $\boldsymbol{A}^{-1}\boldsymbol{B}$。还要注意，两个矩阵乘积之逆等于两个矩阵逆的乘积，即

$$(\boldsymbol{A}\boldsymbol{B})^{-1} = \boldsymbol{B}^{-1}\boldsymbol{A}^{-1}$$

矩阵 $\boldsymbol{A}$ 的秩定义为 $\boldsymbol{A}$ 中最大的非奇异方阵的阶数。假设矩阵 $\boldsymbol{A}$ 为（$m \times n$）矩阵，其中 $n \geqslant m$，如果 $\boldsymbol{A}$ 中最大的子方阵（$m \times m$）是非奇异的，也就是（$m \times m$）行列式非0，则 $\boldsymbol{A}$ 的秩就是 m。可以用 MATLAB 指令 rank（$\boldsymbol{A}$）求取矩阵的秩，如：

```
>> A=[ 1,2,1,0; 1,0,0,1; 2,1,0,1]
```

```
A =
      1     2     1     0
      1     0     0     1
      2     1     0     1
>> rank (A)
ans =
      3
```

每个方阵都有被称为特征值的标量参数，并且每个特征值有其对应的矢量，称为特征矢量。如果如下等式成立，则矩阵 $\boldsymbol{A}$ 就有特征值 λ 和特征矢量 $\boldsymbol{x}$：

$$\boldsymbol{Ax} = \lambda\boldsymbol{x}$$

此式也可以表示为

$$(\boldsymbol{A} - \lambda\boldsymbol{I})\boldsymbol{x} = 0$$

要找出矩阵 $\boldsymbol{A}$ 的特征值 λ 及其对应的特征矢量 $\boldsymbol{x}$ 就需要求解这个齐次方程组。$(\boldsymbol{A} - \lambda\boldsymbol{I})$ 被称为特征矩阵，如果该矩阵行列式的值为 0，则齐次方程组具有非平凡解。例如，有如下 (2×2) 矩阵 $\boldsymbol{A}$

$$\boldsymbol{A} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$$

如果其特征矩阵行列式的值为 0，有

$$|\boldsymbol{A} - \lambda\boldsymbol{I}| = \begin{vmatrix} 1-\lambda & 0 \\ 1 & 1-\lambda \end{vmatrix} = (1-\lambda)^2 = 0 \Rightarrow \lambda_1 = 1, \lambda_2 = 1$$

在 MATLAB 中用 eig（$\boldsymbol{A}$）就可以求得 λ 的值，可以将特征值组成的输出矢量赋给一个变量，如 m = eig（$\boldsymbol{A}$）。

指令［$\boldsymbol{Z}$，$\boldsymbol{D}$］= eig（$\boldsymbol{A}$）生成一个特征值构成的对角矩阵 $\boldsymbol{D}$ 和一个全矩阵 $\boldsymbol{Z}$，$\boldsymbol{Z}$ 中的列就是相应的特征矢量，例如：

```
>> B
B =
     1     3     5
     2     0     4
     1     5     6
>> m = eig (B)
m =
    9.6540
   -0.2596
```

```
  -2.3944
>> [Z, D] =eig (B)
Z =

   0.5587    0.8176    0.0067
   0.4136    0.3791   -0.8595
   0.7189   -0.4334    0.5111

D =

    9.6540         0         0
         0   -0.2596         0
         0         0   -2.3944
```

C.2 矩阵分解

矩阵 $\boldsymbol{A}$ 的 LU 分解就是找出一个下三角形矩阵 $\boldsymbol{L}$ 和一个上三角形矩阵 $\boldsymbol{U}$，使得 $\boldsymbol{A}=\boldsymbol{LU}$，也就是 $\boldsymbol{LU}=\boldsymbol{A}$，即

$$\begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix}\begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}=\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

下面介绍几种求矩阵 $\boldsymbol{L}$ 和 $\boldsymbol{U}$ 的方法。

1. 利用高斯消元法求 $\boldsymbol{L}$ 和 $\boldsymbol{U}$ 矩阵

利用第 4 章介绍的基本高斯消元法可将原始矩阵 $\boldsymbol{A}$ 转化为上三角形矩阵 $\boldsymbol{U}$。同时，用如下所示的算法，将单位矩阵适当位置上的元素值设定为高斯消元法中的乘数的负值，就可以构建出对角线元素为 1 的下三角形矩阵 $\boldsymbol{L}$。其算法为

输入

$\boldsymbol{A}$	需分解的（$n\times n$）矩阵
n	$\boldsymbol{A}$ 的维数

初始化

$\boldsymbol{L}=\boldsymbol{I}$	（$n\times n$）单位矩阵
$\boldsymbol{U}=\boldsymbol{A}$	

运算

```
For k = 1 to n - 1
        For i = k + 1 to n                    对矩阵 U 第 k 行之后的每一行操作
            m(i,k) = - U(i,k)/U(k,k)
                For j = k to n                转化第 i 行
                    U(i,j) = U(i,j) + m(i,k) * U(k,j)
                End
            L(i,k) = - m(i,k)                 更新 L 矩阵
        End
End
    L                                         所求得的下三角形矩阵
    U                                         所求得的上三角形矩阵
```

除了高斯消元法之外，另外还有两种直接 LU 分解方法，一种是 Doolittle 分解法，其矩阵 **L** 的对角线元素为 1；另一种是对称矩阵的 Cholesky 分解，其上三角形矩阵 **U** 为下三角形矩阵 **L** 的转置。

2. Doolittle 分解

假设有（3×3）矩阵 **A**，求其 **L** 和 **U** 矩阵，其中矩阵 **L** 的对角线元素为 1，即

$$\underset{\boldsymbol{L}}{\begin{bmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{bmatrix}} \underset{\boldsymbol{U}}{\begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}} = \underset{\boldsymbol{A}}{\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{33} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}}$$

首先，第一步可以得到 $u_{11} = a_{11}$，并由此求得 **U** 第一行和 **L** 第一列其余元素；第二步求取 u_{22}，并求得 **U** 第二行和 **L** 第二列的其余因素；如此进行下去，就可以得到 **U** 和 **L** 的所有元素。以下是该算法的总结，其中 MATLAB 的冒号 “:” 用于表示行和列。

```
输入
    A                           需分解的（n×n）矩阵
    n                           A 的维数
初始化
    U = zero (n)                U 初始化为（n×n）的零矩阵
    L = I (n)                   L 初始化为单位矩阵
运算
For k = 1 to n
```

$U(k,k) = A(k,k) - L(k,1:k-1) * U(1:k-1,j)$

For $j = k+1$ to n

$U(k,j) = A(k,j) - L(k,1:k-1) * U(1:k-1,j)$

$L(j,k) = (A(j,k) - L(j,1:k-1) * U(1:k-1,k))/U(k,k)$

End

End

---

3. Cholesky 分解

如果矩阵 $\boldsymbol{A}$ 为对称矩阵，那么 Cholesky 分解是一种方便的 LU 分解方法，其上三角形矩阵 $\boldsymbol{U}$ 为下三角形矩阵 $\boldsymbol{L}$ 的转置矩阵，也就是 $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^{\mathrm{T}}$。Cholesky 分解的基本步骤概括如下：

输入

$\boldsymbol{A}$　　($n \times n$) 对称矩阵，即 $\boldsymbol{A} = \boldsymbol{A}^{\mathrm{T}}$

n　　$\boldsymbol{A}$ 的维数

初始化

$\boldsymbol{L} = \boldsymbol{I}(n)$　　($n \times n$) 单位矩阵

运算

For $k = 1$ to n

$x = L(k,1:k-1)$　　$\boldsymbol{L}$ 矩阵第 k 行的第 $1 \sim k-1$ 列

$L(k,k) = \sqrt{A(k,k) - xx^{\mathrm{T}}}$

For $j = k+1$ to n

$y = L(j,1:k-1)$　　$\boldsymbol{L}$ 矩阵第 j 行的第 $1 \sim k-1$ 列

END

END

转置

$\boldsymbol{U} = \boldsymbol{L}^{\mathrm{T}}$

MATLAB 有几条很方便的 LU 分解指令。函数 lu 可用于方阵 $\boldsymbol{A}$ 的 LU 分解，该函数的调用 [$\boldsymbol{L}$, $\boldsymbol{U}$] = lu ($\boldsymbol{A}$) 返回上三角形矩阵 $\boldsymbol{U}$ 以及下三角形矩阵 $\boldsymbol{L}$，使得 $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{U}$。例如：

```
>> a =[4,12,8,4;1,6,25,9;2,9,30,30;3,11,20,18]
a =

     4    12     8     4
```

```
     1     6    25     9
     2     9    30    30
     3    11    20    18

>>
>>[L,U] =lu (a)
L =

    1.0000         0         0         0
    0.2500    1.0000         0         0
    0.5000    1.0000    1.0000         0
    0.7500    0.6667   -0.4444    1.0000

U =

    4.0000   12.0000    8.0000    4.0000
         0    3.0000   23.0000    8.0000
         0         0    3.0000   20.0000
         0         0         0   18.5556
```

Cholesky 分解的 MATLAB 函数是 chol，由于待分解的矩阵为对称矩阵，如下所示，该函数只使用矩阵 **A** 的对角线元素和上三角部分的元素。要注意，Cholesky 分解只适用于正定矩阵，如果对称矩阵 **A** 满足 $\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x}>0$，则 **A** 为正定矩阵，**A** 的所有特征值为正，对角线上所有元素也为正。一般，如果矩阵 **A** 是非奇异的、对角占优的（$|a_{ii}|>\sum_{j\neq i}|a_{ij}|$），并且，对角线上的元素都为正（即 $a_{ii}>0$，$i=1,\ldots,n$），那么 **A** 为正定矩阵。例如：

```
>> A =[1,4,5;4,20,40;5,40,128]
A =

     1     4     5
     4    20    40
     5    40   128

>>
>> chol (A)
ans =

    1.0000    4.0000    5.0000
```

```
0    2.0000   10.0000
0         0    1.7321
```

有关线性代数部分内容的简单复习就到此为止。线性代数方面的教科书非常多，我们鼓励有兴趣的同学自学更多的内容。

附录 D　微分方程的解析解

本附录概述常微分方程和偏微分方程解析解的求法，并不是对这些求解方法进行完整的论述，只是一个方便学生学习的简明回顾。我们假定同学们已经学过微积分课程中的常微分方程，那门课程对于常微分方程解析解的求法有详细论述。但是，本科生对于偏微分方程的求解可能还比较陌生，同学们最好学习一下本附件的内容。有关这些内容的详细论述请参阅 Boyce 和 DePrima（2001）、O'Neil（2003）以及 Zill 和 Cullen（2000）等人的著作。

本附录中的每种方法都有举例说明，可能的话，还使用 MATLAB 的符号数学工具箱（Symbolic Math Toolbox）进行求解，使同学们有机会练习使用 MATLAB 的符号逻辑功能。

D.1　一阶常微分方程

一阶常微分方程的标准形式为

$$M(t,y)\,\mathrm{d}t + N(t,y)\,\mathrm{d}y = 0 \tag{D.1}$$

也可以表示为

$$\frac{\mathrm{d}y}{\mathrm{d}t} = f(t,y) \tag{D.2}$$

在下面的论述中，这两种形式都会用到。本附录以及本书的其他部分都用符号 y 表示常微分方程的因变量，用 t 表示自变量。不过，如果需要的话，例如自变量表示的是空间信息而不是时间，那么也用符号 x 表示自变量。

D.1.1　变量可分离的微分方程

如果方程（D.2）中的函数 $f(t,\ y)$ 可以写为

$$f(t,y) = g(t)h(y) \tag{D.3}$$

式中　$g(t)$ ——t 的函数；

$h(y)$ ——y 的函数。

则微分方程

$$\frac{\mathrm{d}y}{\mathrm{d}t} = g(t)h(y) \tag{D.4}$$

具有可以分离的变量。这种方程的求解是先把方程化为

$$\frac{dy}{h(y)} = g(t)dt \tag{D.5}$$

再两边求积分

$$\int \frac{dy}{h(y)} = \int g(t)dt \tag{D.6}$$

例如，对于以下简单的线性微分方程

$$\frac{dy}{dt} = \lambda y \tag{D.7}$$

分离变量并积分，可得

$$\int \frac{dy}{y} = \int \lambda dt$$
$$\ln y = \lambda t + \ln c \tag{D.8}$$

为了简化最终解的形式，这里的积分常数用了对数的形式。对上式两边进行指数运算，可得最终解的形式为：

$$y = ce^{\lambda t} \tag{D.9}$$

下面将会看到，这种形式的解还可能成为其他微分方程的解。

例 D.1　求解微分方程

$$\frac{dy}{dt} = \frac{y}{1+t}$$

解：

分离变量并两边求积分，有

$$\int \frac{dy}{y} = \int \frac{dt}{1+t}$$

得到解

$$\ln(y) = \ln(1+t) + \ln(c)$$

两边求指数，并简化解的形式，有

$$y = c(1+t)$$

MATLAB 的求解指令为

```
>> Y=dsolve ('Dy=y/(1+t)')
Y=C1 * (1+t)
```

D.1.2　齐次方程

如果方程（D.1）中的 $M(t, y)$ 和 $N(t, y)$ 是具有相同次方的齐次函

数，则变换两个变量中的一个变量，如 $y = vt$，就可以将方程化为变量可分离型的方程。

当且仅当 $f(mt,my) = m^k f(t,y)$ 时（m 为常数），函数 $f(t,y)$ 才是 t 和 y 的 k 次方齐次函数。

例 D.2 解微分方程

$$(t^2 - ty + y^2)\mathrm{d}t - ty\mathrm{d}y = 0$$

解：

将 y 变换为 vt，则

$$(t^2 - t^2v + t^2v^2)\mathrm{d}t - t^2v(v\mathrm{d}t + t\mathrm{d}v) = 0$$

除以 t^2，简化为

$$(1 - v + v^2)\mathrm{d}t - v(v\mathrm{d}t + t\mathrm{d}v) = 0$$

重排式中各项并分离变量，有

$$\frac{\mathrm{d}t}{t} + \frac{v\mathrm{d}v}{v-1} = 0$$

由于

$$\frac{v}{v-1} = 1 + \frac{1}{v-1}$$

微分方程可以化为

$$\frac{\mathrm{d}t}{t} + \left[1 + \frac{1}{v-1}\right]\mathrm{d}v = 0$$

得到解（为了与 MATLAB 的常数选择一致，积分常数取为 $-c_1$）

$$\ln t + v + \ln(v-1) = -c_1$$

简化为

$$t(v-1)\mathrm{e}^v = \mathrm{e}^{-c_1}$$

代回原始变量 y，则

$$t\left(\frac{y}{t} - 1\right)\mathrm{e}^{y/t} = \mathrm{e}^{-c_1}$$

此解中 y 是隐式的。我们再用如下 MATLAB 的 solve 指令求 y 的显式解：

```
>> YY = solve('t * (y/t - 1) * exp(y/t) = exp( - c1)','y')
YY =
(lambertw(1/t * exp( - c1 - 1)) + 1) * t
```

也可以用 MATLAB 指令直接求解微分方程，则

```
>> Y = dsolve('t * y * Dy = (t^2 - t * y + y^2)')
Y =
t + exp( - lambertw(1/exp(C1) * exp( - 1)/t) - C1 - 1)
>> simplify(Y)
ans =
(lambertw(1/t * exp( - C1 - 1)) + 1) * t
```

以上两种方法得到的两个解相同。y 的显式解中用到了 Lambert-W 函数，该函数的定义为 w = lambertw（x），是方程 w * exp（w） = x 的解。

D.1.3　全微分方程

如果存在函数 $F(t,y) = c$，其全微分正好等于 $M(t,y)\mathrm{d}t + N(t,y)\mathrm{d}y$，那么，方程（D.1）就是一个全微分方程。这时，以下关系式成立：

$$\frac{\partial M}{\partial y} = \frac{\partial N}{\partial t} \tag{D.10}$$

并且

$$\frac{\partial F}{\partial t} = M(t,y) \tag{D.11}$$

$$\frac{\partial F}{\partial y} = N(t,y) \tag{D.12}$$

全微分方程（D.1）的解就是 F，可以通过方程（D.11）对 t 的积分

$$F(t,y) = \int M\mathrm{d}t + h(y) \tag{D.13}$$

以及方程（D.12）对 y 的积分

$$F(t,y) = \int N\mathrm{d}y + g(t) \tag{D.14}$$

求得 F 的解。其中，$h(y)$ 和 $g(t)$ 可以通过比较（D.13）和（D.14）两式来确定。

例 D.3　求解微分方程

$$3t(ty - 2)\mathrm{d}t + (t^3 + 2y)\mathrm{d}y = 0$$

解：

首先验证这是一个全微分方程：

$$\frac{\partial M}{\partial y} = 3t^2 = \frac{\partial N}{\partial t}$$

利用式（D.11）和式（D.12），有：

$$\frac{\partial F}{\partial t} = M = 3t(ty-2)$$

$$\frac{\partial F}{\partial y} = N = t^3 + 2y$$

利用式（D.13）和式（D.14）可以求解此全微分方程，即将以上第一个等式对 t 求积分，第二个等式对 y 求积分，得到：

$$F = t^3y - 3t^2 + h(y)$$

$$F = t^3y + y^2 + g(t)$$

要使这两个方程式的右边相等，必须有

$$h(y) = y^2$$

$$g(t) = -3t^2$$

于是，函数 F 为

$$F = t^3y - 3t^2 + y^2$$

微分方程的解就是

$$t^3y - 3t^2 + y^2 = c$$

下面用 MATLAB 指令求解：

```
>> y = dsolve('Dy = 3 * t * (t * y - 2)/(t^3 + 2 * y)')
Warning: Explicit solution could not be found.
```

可见，MATLAB 不能求得该方程的显式解。

D.1.4　线性微分方程和积分因子

以下方程是一个线性非齐次常微分方程（有关常微分方程的完整定义请参见 7.2 节）

$$\frac{dy}{dt} + P(t)y = f(t) \tag{D.15}$$

要求解该微分方程，将等式两边同乘以积分因子

$$[e^{\int P(t)dt}]$$

得到

$$e^{\int P(t)dt}\frac{dy}{dt} + e^{\int P(t)dt}P(t)y = e^{\int P(t)dt}f(t) \tag{D.16}$$

此式左边是 $[e^{\int P(t)dt}y]$ 对 t 的求导，因此有

$$\frac{d}{dt}[e^{\int P(t)dt}y] = e^{\int P(t)dt}f(t) \tag{D.17}$$

将此式两边求积分，可得

$$e^{\int P(t)\mathrm{d}t}y = c + \int e^{\int P(t)\mathrm{d}t}f(t)\,\mathrm{d}t \tag{D.18}$$

重排此式，得到方程的解：

$$y = ce^{-\int P(t)\mathrm{d}t} + e^{-\int P(t)\mathrm{d}t}\int e^{\int P(t)\mathrm{d}t}f(t)\,\mathrm{d}t \tag{D.19}$$

例 D.4　求解微分方程

$$\frac{\mathrm{d}y}{\mathrm{d}t} - t^2 y = t^3 e^t$$

解：

先求积分因子

$$e^{\int P(t)\mathrm{d}t} = e^{\int -t^2\mathrm{d}t} = e^{-t^3/3}$$

用此积分因子乘以微分方程的两边，可得

$$e^{-t^3/3}\frac{\mathrm{d}y}{\mathrm{d}t} - e^{-t^3/3}t^2 y = e^{-t^3/3}t^3 e^t$$

注意，该等式左边是 $[e^{-t^3/3}y]$ 的导数，因此有

$$\frac{\mathrm{d}}{\mathrm{d}t}[e^{-t^3/3}y] = e^{-t^3/3}t^3 e^t$$

将此式两边求积分

$$e^{-t^3/3}y = c + \int e^{-t^3/3}t^3 e^t\mathrm{d}t$$

将此式两边除以 $e^{-t^3/3}$，有

$$y = ce^{t^3/3} + \int t^3 e^t\mathrm{d}t$$

利用分部积分法 ($\int u\mathrm{d}v = uv - \int v\mathrm{d}u$) 求该积分，可得方程的解：

$$y = ce^{t^3/3} + t^3e^t - 3t^2e^t + 6te^t - 6e^t$$

下面用 MATLAB 指令求解该微分方程：

```
>> dsolve('Dy - t^2 * y = t^3 * exp(t)')
ans =
(Int(t^3 * exp((1 - 1/3 * t^2) * t),t) + C1) * exp(1/3 * t^3)
```

结果就是

$$y = ce^{t^3/3} + \int t^3 e^t\mathrm{d}t$$

进一步求解，需要用以下指令分离常数项，并求积分：

```
>> syms t C1
>> Int(t^3 * exp((1 - 1/3 * t^2) * t) * exp(1/3 * t^3)) + C1 * exp
(1/3 * t^3)
ans =
t^3 * exp(t) - 3 * t^2 * exp(t) + 6 * t * exp(t) - 6 * exp(t) + C1 * exp
(1/3 * t^3)
```

可见，MATLAB 的解与前面得到的解析解相同。

D.1.5 非线性微分方程和积分因子

如下伯努利方程（Bernoulli Equation）是非线性非齐次常微分方程：

$$\frac{dy}{dt} + P(t)y = f(t)y^n \tag{D.20}$$

其中 $n \neq 1$。用适当的变量代换可以将此方程化为（D.15）形式的线性方程，然后可以利用积分因子求解。

为了确定变量代换式，先将式（D.20）两边同乘以 y^{-n}

$$y^{-n}\frac{dy}{dt} + P(t)y^{1-n} = f(t) \tag{D.21}$$

注意，y^{1-n}的微分为 $(1-n)\ y^{-n}$，即

$$\frac{d(y^{1-n})}{dt} = (1-n)y^{-n}\frac{dy}{dt} \tag{D.22}$$

用以下变换

$$z = y^{1-n} \tag{D.23}$$

简化（D.22）方程，得到

$$\frac{dz}{dt} = (1-n)y^{-n}\frac{dy}{dt} \tag{D.24}$$

并用这个代换式，将伯努利方程化为

$$\left(\frac{1}{1-n}\right)\frac{dz}{dt} + P(t)z = f(t) \tag{D.25}$$

可见，这就是方程（D.15）的形式，可以用 D.1.4 节的积分因子的方法求解。

D.2 高阶常微分方程

n 阶线性常微分方程的通式可以表示为

$$b_n(t)\frac{d^n y}{dt^n} + b_{n-1}(t)\frac{d^{n-1}y}{dt^{n-1}} + \cdots + b_1(t)\frac{dy}{dt} + b_0(t)y = R(t) \tag{D.26}$$

如果 $R(t)=0$，则方程为齐次方程。如果 $R(t)\neq 0$，则为非齐次方程。系数 $\{b_i | i=n,\cdots,1\}$ 为 t 的函数时，称为变系数；为常量时，则称为常系数。如果方程中自变量不显式出现，则微分方程是自治的。

D.2.1　常系数齐次线性微分方程

如下是三阶齐次线性自治微分方程

$$b_3\frac{d^3y}{dt^3}+b_2\frac{d^2y}{dt^2}+b_1\frac{dy}{dt}+b_0y=0 \tag{D.27}$$

该方程可以写成微分算子 D 的形式

$$b_3D^3y+b_2D^2y+b_1Dy+b_0y=0 \tag{D.28}$$

其中，$D\equiv\frac{d}{dt}$。式中 y 乘在每一项之后，可以把它提取出来

$$(b_3D^3+b_2D^2+b_1D+b_0)y=0 \tag{D.29}$$

括号中的项是微分算子的线性组合，整个可以看作是对 y 运算的一个微分算子。要使式（D.29）对任意的 y 值都成立，则括号中的项必须等于0，即

$$(b_3D^3+b_2D^2+b_1D+b_0)=0 \tag{D.30}$$

该式称为微分方程的特征方程，这是一个 n 次多项式（此例中 $n=3$）有 n 个根 $(\lambda_1,\lambda_2,\lambda_3)$。这些根称为特征根，可以是互异的实根、重实根、复数根、或者这些不同形式根的组合。复数根总是以共轭对的形式出现。

如果是互异的实根，即 $\lambda_1\neq\lambda_2\neq\lambda_3$，利用（D.9）解的形式，可以用这些根构建微分方程的解

$$y=c_1e^{\lambda_1t}+c_2e^{\lambda_2t}+c_3e^{\lambda_3t} \tag{D.31}$$

每个特征根构成的那部分解称为特征函数。

如果有重实根，即 $\lambda_1=\lambda_2\neq\lambda_3$，则解可以写为

$$y=c_1e^{\lambda_1t}+c_2te^{\lambda_2t}+c_3e^{\lambda_3t} \tag{D.32}$$

如果是一对复数根和一个实根，即

$$\lambda_1=\alpha+\beta i,\quad \lambda_2=\alpha-\beta i,\quad \lambda_3=\text{实数},\quad \text{其中 } i=\sqrt{-1}$$

则解为

$$y=c_1e^{(\alpha+\beta i)t}+c_2e^{(\alpha-\beta i)t}+c_3e^{\lambda_3t} \tag{D.33}$$

利用欧拉恒等式

$$e^{\alpha+\beta i}=e^{\alpha}(\cos\beta+i\sin\beta)$$
$$e^{\alpha-\beta i}=e^{\alpha}(\cos\beta-i\sin\beta) \tag{D.34}$$

可将式（D.33）转化为

$$y=c_4e^{\alpha t}\cos(\beta t)+c_5e^{\alpha t}\sin(\beta t)+c_3e^{\lambda_3t} \tag{D.35}$$

其中两个新的系数为

$$c_4 = c_1 + c_2$$
$$c_5 = i(c_1 - c_2)$$

例 D.5 求如下微分方程的解

$$\frac{d^3y}{dt^3} - 9\frac{d^2y}{dt^2} + 26\frac{dy}{dt} - 24y = 0$$

解:

引入微分算子

$$D^3y - 9D^2y + 26Dy - 24y = 0$$

得到特征方程

$$(D^3 - 9D^2 + 26D - 24) = 0$$

因式分解求其根

$$(D-2)(D-4)(D-3) = 0$$

因此，该特征方程的根为

$$\lambda_1 = 2,\quad \lambda_2 = 4,\quad \lambda_3 = 3$$

解的构成如下

$$y = c_1 e^{2t} + c_2 e^{4t} + c_3 e^{3t}$$

将此解代入原微分方程可以验证其正确性。

下面用 MATLAB 指令求解

```
>> Y = dsolve('D3y - 9 * D2y + 26 * Dy - 24 * y = 0')
Y =
C1 * exp(2 * t) + C2 * exp(4 * t) + C3 * exp(3 * t)
```

如果给定微分方程的初始条件，就可以求得积分常数。例如，$y(0) = 1$，$\left.\frac{dy}{dt}\right|_0 = 2$，$\left.\frac{d^2y}{dt^2}\right|_0 = 3$，则用 MATLAB 指令，通过代数运算就可以求得积分常数:

```
>> Y = dsolve('D3y - 9 * D2y + 26 * Dy - 24 * y = 0','y(0) = 1','Dy(0)
= 2','D2y(0) = 3')
Y =
1/2 * exp(2 * t) - 1/2 * exp(4 * t) + exp(3 * t)
```

D.2.2 常系数非齐次线性微分方程

如下方程是非齐次常微分方程:

$$\frac{d^3y}{dt^3} - 9\frac{d^2y}{dt^2} + 26\frac{dy}{dt} - 24y = e^t \tag{D.36}$$

这类方程的解由一个通解和一个特解组成：

$$y = y_c + y_p \tag{D.37}$$

通解 y_c 是与（D.36）方程相对应的齐次方程

$$\frac{d^3y}{dt^3} - 9\frac{d^2y}{dt^2} + 26\frac{dy}{dt} - 24y = 0$$

的解，该齐次方程的解在例 D.5 中已求得，为 $y_c = c_1e^{2t} + c_2e^{4t} + c_3e^{3t}$。特解 y_p 对应于方程的非齐次部分，即式（D.36）的右边。考察该等式的右边，可以推测 e^t 来自于方程式的一个根 $\lambda_4 = 1$，也就是

$$(D-1)R(t) = 0 \tag{D.38}$$

用（D－1）乘以式（D.36）的两边，使等式右边为0：

$$(D-1)(D^3 - 9D^2 + 26D - 24)y = (D-1)e^t = 0 \tag{D.39}$$

此式为齐次微分方程，特征根为

$$\lambda_1 = 2, \quad \lambda_2 = 4, \quad \lambda_3 = 3, \quad \lambda_4 = 1$$

其解为

$$y = c_1e^{2t} + c_2e^{4t} + c_3e^{3t} + Ce^t \tag{D.40}$$

此解的最后一项为特解部分，其常数用 C 表示，C 的取值必须使这个特解满足式（D.36）。为了求取 C 的值，将特解 y_p 的一阶、二阶以及三阶导数代入式（D.36），求得 $C = -1/6$，于是，整个解为

$$y = c_1e^{2t} + c_2e^{4t} + c_3e^{3t} - \frac{1}{6}e^t \tag{D.41}$$

以下利用 MATLAB 指令求解该微分方程：

```
>> Y = dsolve ('D3y - 9 * D2y + 26 * Dy - 24 * y = exp (t) ')
Y =
-1/6 * exp (t) + C1 * cxp (2 * t) + C2 * exp (4 * t) + C3 * exp (3 * t)
```

例 D.6 求以下非齐次微分方程的解

$$\frac{d^3y}{dt^3} - 6\frac{d^2y}{dt^2} + 11\frac{dy}{dt} - 6y = \sin t + \cos t$$

解：

引入微分算子

$$D^3y - 6D^2y + 11Dy - 6y = 0$$

得到特征方程

$$(D^3 - 6D^2 + 11D - 6) = 0$$

因式分解并求根

$$(D-1)(D-2)(D-3) = 0$$

因此，特征方程的根为

$$\lambda_1 = 1, \quad \lambda_2 = 2, \quad \lambda_3 = 3$$

微分方程的通解为

$$y_c = c_1 e^t + c_2 e^{2t} + c_3 e^{3t}$$

特解为

$$y_p = C_1 \cos t + C_2 \sin t$$

为了计算 C_1 和 C_2 的值，将特解 y_p 的一阶、二阶以及三阶导数代入微分方程，求得

$$C_1 = -\frac{1}{10} \quad C_2 = \frac{1}{10}$$

于是，完整的解是

$$y_c = c_1 e^t + c_2 e^{2t} + c_3 e^{3t} - \frac{1}{10}\cos t + \frac{1}{10}\sin t$$

利用 MATLAB 指令求解该微分方程：

```
>> Y = dsolve('D3y - 6 * D2y + 11 * Dy - 6 * y = cos(t) + sin(t)')
Y =
 -1/10 * cos(t) +1/10 * sin(t) + C1 * exp(t) + C2 * exp(2 * t) + C3 *
exp(3 * t)
```

如果给定微分方程的初始条件，可以求得通解的积分常数。例如，$y(0) = 1$，$\left.\frac{dy}{dt}\right|_0 = 2$，$\left.\frac{d^2y}{dt^2}\right|_0 = 3$，则用 MATLAB 指令，通过代数运算就可以求得积分常数：

```
>> Y = dsolve('D3y - 6 * D2y + 11 * Dy - 6 * y = cos(t) + sin(t)',
'y(0) =1','Dy(0) =2','D2y(0) =3')
Y =
 -1/10 * cos(t) +1/10 * sin(t) +7/5 * exp(2 * t) -3/10 * exp(3 * t)
```

D.3 变量可分离偏微分方程

偏微分方程的分类见 8.2 节，有一阶、二阶、三阶和四阶的线性或非线性、

齐次或非齐次方程。工程和物理学中最常见的是二阶偏微分方程。因此，我们主要介绍二阶线性偏微分方程的解析解。

偏微分方程的解可以包含任意函数或者包含无限个任意常数。偏微分方程的通解可以定义为包含 n 个任意函数的解。

D.3.1　扩散方程

具有可分离变量的偏微分方程的典型例子就是一维非稳态扩散方程：

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2} \tag{D.42}$$

如果 u 为浓度，α 为扩散系数，则该方程表示某种溶液通过渗透膜的扩散机制。如果 u 为温度，α 为温度扩散率，则此方程可以描述平板中的能量扩散。

为了便于讨论，假设膜的厚度有限，为 L；并且初始时膜内溶液的浓度为距离的函数，设为 $f(x)$。实验开始时，假设膜被浸入浓度保持为 0 的大容量液体中，膜的两个表面都暴露在液体之中。因此，方程（D.42）的初始条件和边界条件为

$$\begin{aligned} &\text{当 } t = 0, \text{且 } 0 \leqslant x \leqslant L \text{ 时}, u(x,0) = f(x) \\ &\text{当 } t > 0 \text{ 时}, u(0,t) = 0 \text{ 并且 } u(L,t) = 0 \end{aligned} \tag{D.43}$$

假设偏微分方程的一个解 u 可以表示为两个函数 $X(x)$ 和 $T(t)$ 的乘积，这里 $X(x)$ 只是 x 的函数，$T(t)$ 只是 t 的函数，即：

$$u(x,t) = X(x)T(t) \tag{D.44}$$

将此式代入式（D.42），得到

$$X(x)T'(t) = \alpha X''(x)T(t) \tag{D.45}$$

此方程的变量可以分离，即

$$\frac{X''(x)}{X(x)} = \frac{T'(t)}{\alpha T(t)} \tag{D.46}$$

此式左边仅为 x 的函数，右边仅为 t 的函数，所以，要使两边相等，它们的值必须为常数，即

$$\frac{X''(x)}{X(x)} = \frac{T'(t)}{\alpha T(t)} = -\lambda^2 \tag{D.47}$$

此处常数设为 $-\lambda^2$，其原因一会儿就清楚了。现在，将式（D.47）分成两个常微分方程：

$$\begin{aligned} X'' + \lambda^2 X = 0 \\ T' + \alpha\lambda^2 T = 0 \end{aligned} \tag{D.48}$$

用 D.2 节的方法可以求解这两个方程：

$$X = c_1\cos(\lambda x) + c_2\sin(\lambda x)$$

$$T = c_3 e^{-\alpha\lambda^2 t} \tag{D.49}$$

利用边界条件

$$\left.\begin{aligned} u(0,t) &= X(0)T(t) = 0 \\ u(L,t) &= X(L)T(t) = 0 \end{aligned}\right\} 当\ t > 0\ 时 \tag{D.50}$$

可知 $X(0) = 0$ 并且 $X(L) = 0$，代入式（D.49）的第一个等式，可得 $c_1 = 0$，并且

$$X(L) = c_2 \sin(\lambda L) = 0$$

对于非平凡解，c_2 和 λ 不能为0，因此有

$$\sin(\lambda L) = 0$$

这就意味着 $\lambda L = n\pi$，$n = 1, 2, 3, \cdots, \infty$，$\lambda$ 即为

$$\lambda = \frac{n\pi}{L}, n = 1,2,3,\cdots,\infty \tag{D.51}$$

$X(x)$ 和 $T(t)$ 的解就可以表示为 $X_n(x)$ 和 $T_n(t)$：

$$\left.\begin{aligned} X_n(x) &= c_{2_n} \sin\left(\frac{n\pi x}{L}\right) \\ T_n(t) &= c_{3_n} e^{-\alpha\left(\frac{n^2\pi^2}{L^2}\right)t} \end{aligned}\right\} 其中\ n = 1,2,3,\cdots,\infty \tag{D.52}$$

代入式（D.44），可得

$$u_n = A_n e^{-\alpha\left(\frac{n^2\pi^2}{L^2}\right)t} \sin\left(\frac{n\pi x}{L}\right) \quad 其中\ n = 1,2,3,\cdots,\infty \tag{D.53}$$

这里，用新的常数 A_n 代替了 c_{2_n} 和 c_{3_n} 两个常数的乘积，它是 n 的函数。根据叠加原理，如果在一定范围内，u_1，u_2，$\cdots$，u_n 为某个线性齐次偏微分方程的解，则这些解的任意线性组合也是该方程在这个范围内的解。为了求取满足任意边界条件的解，考虑所有解的无限级数组合

$$u(x,t) = \sum_{n=1}^{\infty} u_n = \sum_{n=1}^{\infty} A_n e^{-\alpha\left(\frac{n^2\pi^2}{L^2}\right)t} \sin\left(\frac{n\pi x}{L}\right) \tag{D.54}$$

应用初始条件：$t = 0$ 时，对于 $0 \leqslant x \leqslant L$，有 $u(x, 0) = f(x)$，则

$$u(x,0) = f(x) = \sum_{n=1}^{\infty} A_n \sin\left(\frac{n\pi x}{L}\right) \tag{D.55}$$

为了求取 A_n，将此式两边同乘以 $\sin\left(\frac{m\pi x}{L}\right)$，其中 m 为任意整数 $m = 1, 2, 3, \cdots, \infty$，并且将两边在区间（0，$L$）上求积分，则：

$$\int_0^L f(x)\sin\left(\frac{m\pi x}{L}\right)dx = \sum_{n=1}^{\infty} A_n \int_0^L \left(\sin\left(\frac{m\pi x}{L}\right)\sin\left(\frac{n\pi x}{L}\right)\right)dx \tag{D.56}$$

在区间（0，L）上，权重函数为1时正弦函数是正交函数，因此

$$\int_0^L \left(\sin\left(\frac{m\pi x}{L}\right) \sin\left(\frac{n\pi x}{L}\right) \right) dx = 0 \quad 当 n \neq m 时 \tag{D.57}$$

但是，当 $n = m$ 时，应用积分表，如下所示，可以求得该积分的值为 $L/2$：

$$\int_0^L \sin^2\left(\frac{n\pi x}{L}\right) dx = \left[\frac{1}{2}x - \frac{1}{4\frac{n\pi}{L}} \sin\left(\frac{2n\pi x}{L}\right) \right]_0^L = \frac{L}{2} \tag{D.58}$$

利用此式，式（D.56）的加和计算中只剩下一项，因此就可以求解 A_n，即：当 $n = m$ 时

$$A_n = \frac{2}{L}\int_0^L f(x) \sin\left(\frac{n\pi x}{L}\right) dx \tag{D.59}$$

于是，在式（D.43）给定的初始条件和边界条件下，方程（D.42）的解为

$$u(x,t) = \frac{2}{L}\sum_{n=1}^{\infty} \left(\int_0^L f(x) \sin\left(\frac{n\pi x}{L}\right) dx \right) e^{-\alpha\left(\frac{n^2\pi^2}{L^2}\right)t} \sin\left(\frac{n\pi x}{L}\right) \tag{D.60}$$

例 D.7 利用以下数据，求上述膜扩散问题中的浓度分布状况：

$$u(x,0) = 1.0, L = \pi \quad 并且 \alpha = 1$$

解：

将这些值代入式（D.59），得到 A_n 值：

$$A_n = \frac{2}{\pi}\left[\frac{1 - (-1)^n}{n} \right]$$

式（D.54）的解即为

$$u(x,t) = \frac{2}{\pi}\sum_{n=1}^{\infty} \left[\frac{1 - (-1)^n}{n} \right] e^{-n^2 t} \sin(nx)$$

MATLAB 的求解：

在自变量取值范围 $0 < t < 1.5$ 中，下列 MATLAB 程序求解方程并作图：

```
% Evaluating the analytical solution of the one-dimensional
% unsteady-state diffusion equation
clear, clc

L = pi; alpha = 1;
space_steps = 50;
dx = L/space_steps;
time_steps = 15;
dt = 1.5/time_steps;
u = zeros(space_steps,time_steps);
```

```
    u(1:space_steps+1,1)=1;
    syms tt x
    for n=1:2:13
        f(n)=((1-(-1)^n)/n)*exp(-n^2*tt)*sin(n*x);
    end
    sumf=sum(f);
    tt=0;
    for t=1:time_steps+1
          tt=tt+dt;
          x=0;
             for i=2:space_steps
               x=x+dx;
               u(i,t+1)=(2/pi)*eval(sum(f));
          end
    end

    figure(1)
    xx=[0:dx:L];
    plot(xx,u(:,1), xx,u(:,ceil(0.1/dt)+1), xx,u(:,ceil(0.2/dt)
+1),...
          xx,u(:,ceil(0.3/dt)+1), xx,u(:,ceil(0.5/dt)+1),...
          xx,u(:,ceil(1/dt)+1),xx,u(:,ceil(1.5/dt)+1))
    axis tight
    title('One-dimensional unsteady-state diffusion')
    xlabel('Membrane thickness, x'); ylabel('Concentration, y')
    text(L/30, 0.97,'t=0.0'); text(L/7,0.93,'t=0.1');
    text(L/2.2,0.95,'t=0.2'); text(L/2.2,0.88,'t=0.3');
    text(L/2.2,0.72,'t=0.50'); text(L/2.2,0.42,'t=1.00');
    text(L/2.2,0.23,'t=1.50')
```

结果讨论（见图 D.1）

$t=0$ 时的浓度分布曲线为 $u(x,0)=1$，也就是初始条件。$t>0$ 时，与液体接触的膜的外表面，即 $x=0$ 和 $x=L$ 处，有 $u=0$，也就是边界条件。随着 t 的增加，溶液从膜中扩散到周围液体中，使得膜中的溶液浓度下降，并趋于 0。要注意，当 n 为偶数时，系数 $A_n=0$，所以，只需要求级数中的奇数项。另外，当

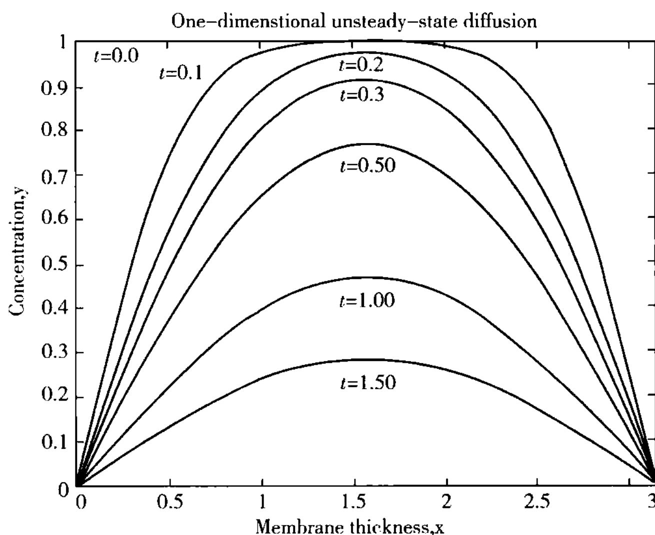


图 D.1 膜扩散问题中浓度分布的 MATLAB 求解结果

$n=13$ 时，级数的值收敛于 $10^{-6}\%$ 之内，这是因为解中有 e^{-n^2t} 项。

D.3.2 位势方程

如下式（D.61）为位势方程，更为人们所熟悉的名称是拉普拉斯方程。可用于模拟力学、电磁场和重力场中的位势，也可用于模拟矩形平板等二维物体的稳态热传导和扩散。

$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}=0 \tag{D.61}$$

如果 u 为浓度，该方程则表示某种溶液在矩形平面上的稳态扩散机制；如果 u 为温度，该方程则描述矩形平板上能量的稳态扩散。

拉普拉斯方程满足变量分离方法的要求。很多种边界条件可以与式（D.61）相结合构成完整的命题。本书第 8 章详细讲述了可以用于这个问题的各种可能的边界条件。

如图 D.2 所示，下面我们求解一个尺寸为 $a\times b$ 的矩形平板上的热传导问题。

温度 T 的拉普拉斯方程为

$$\frac{\partial^2 T}{\partial x^2}+\frac{\partial^2 T}{\partial y^2}=0 \tag{D.62}$$

平板左右两边是理想的绝热体（Neumann 条件），平板下方的温度为 T_0，上方的温度为 x 的函数（Dirichlet 条件）。用数学方法表示这些边界条件如下：

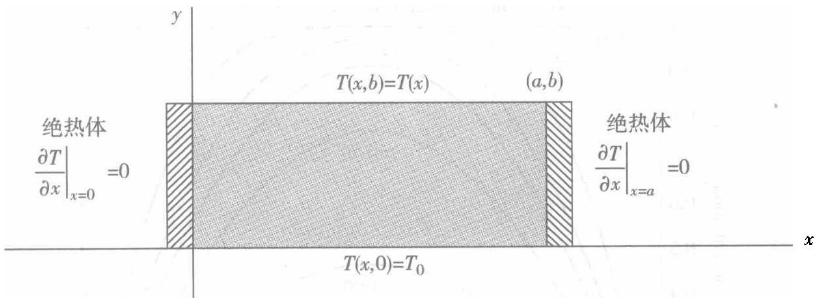


图 D.2 矩形平板上的热传导

$$\left.\frac{\partial T}{\partial x}\right|_{x=0} = 0 \qquad \left.\frac{\partial T}{\partial x}\right|_{x=a} = 0 \qquad 当 0 \leqslant y \leqslant b 时$$
$$T(x,0) = T_0 \quad T(x,b) = T(x) \quad 当 0 < x < a 时 \tag{D.63}$$

为了便于求解，并使解具有普遍意义，用以下等式将变量 T 变换为变量 u:

$$u = \frac{T - T_0}{T_0} \tag{D.64}$$

并且

$$f(x) = \frac{T(x) - T_0}{T_0} \tag{D.65}$$

这样，边界条件就变换成为

$$当 0 \leqslant y \leqslant b 时, \quad \left.\frac{\partial u}{\partial x}\right|_{x=0} = 0 \qquad \left.\frac{\partial u}{\partial x}\right|_{x=a} = 0$$
$$当 0 < x < a 时, \quad u(x,0) = 0 \quad u(x,b) = f(x) \tag{D.66}$$

于是，式（D.62）就与式（D.61）相同。变换后的热传导问题如图 D.3 所示。

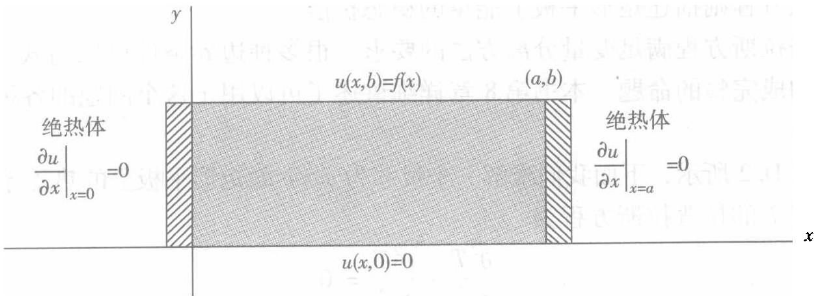


图 D.3 经过变量变换之后矩形平板上的热传导问题

下面是2000 年 Zill 和 Cullen 用式（D.66）边界条件求得的（D.61）方程的解，其中有所改动。

假设 u 是偏微分方程的一个解，并且可以表示为两个函数 $X(x)$ 和 $Y(y)$ 的乘积，这里 $X(x)$ 只是 x 的函数，而 $Y(y)$ 只是 y 的函数。于是：

$$u(x,y) = X(x)Y(y) \tag{D.67}$$

求该式的微分，并代入（D.61）方程，可以分离变量：

$$\frac{X''(x)}{X(x)} = -\frac{Y''(y)}{Y(y)} = -\lambda^2 \tag{D.68}$$

生成两个常微分方程：

$$X'' + \lambda^2 X = 0 \tag{D.69}$$

$$Y'' - \lambda^2 Y = 0 \tag{D.70}$$

第一个方程的解为

$$X = c_1\cos(\lambda x) + c_2\sin(\lambda x) \tag{D.71}$$

第二个方程的解为

$$Y = c_3 e^{\lambda y} + c_4 e^{-\lambda y} \tag{D.72}$$

下面用边界条件求 c_1 到 c_4 这 4 个常数的值。第一个边界条件等效于 $X'(0) = 0$，求式（D.71）的微分，并代入此边界条件，有

$$X'(0) = -c_1\lambda\sin(0) + c_2\lambda\cos(0) = 0 \tag{D.73}$$

因为 $\sin(0) = 0$ 且 $\cos(0) = 1$，因此 c_2 必定为0，于是，式（D.71）可以简化为

$$X = c_1\cos(\lambda x) \tag{D.74}$$

第二个边界条件等效于 $X'(a) = 0$，求式（D.74）的微分，并应用此边界条件，有

$$X'(a) = -c_1\lambda\sin(\lambda a) = 0 \tag{D.75}$$

当 $\lambda = 0$，或者 $\lambda a = n\pi$（即 $\lambda = n\pi/a$），$n = 1, 2, 3, \cdots, \infty$ 时，上式成立。在 $\lambda = 0$，$n = 0$ 的情况下，式（D.69）成为

$$X'' = 0 \tag{D.76}$$

将此式积分两次，可得直线方程：

$$X = c_1 + c_2 x \tag{D.77}$$

应用边界条件 $X'(0) = 0$ 和 $X'(a) = 0$，可知 $X = c_1$，因此式（D.69）的解为

$$\begin{aligned} &\text{当 } n = 0 \text{ 时，} X = c_{1_0}, \\ &\text{并且，当 } n = 1, 2, 3, \cdots, \infty \text{ 时，} X = c_{1_n}\cos\left(\frac{n\pi x}{a}\right), \end{aligned} \tag{D.78}$$

第三个边界条件 $u(x,0) = 0$ 等效于 $Y(0) = 0$，将其应用于式（D.72），有

$$\begin{aligned} Y(0) &= c_3 e^{\lambda 0} + c_4 e^{-\lambda 0} = 0 \\ &= c_3 + c_4 = 0 \end{aligned} \tag{D.79}$$

可见 $\lambda > 0$ 时 $c_4 = -c_3$，式（D.72）可以化为

$$Y = c_3(e^{\lambda y} - e^{-\lambda y}) \tag{D.80}$$

利用以下指数与双曲线正弦函数之间的关系式：

$$\frac{1}{2}(e^{\alpha} - e^{-\alpha}) = \sinh\alpha \tag{D.81}$$

可以将式（D.80）化为

$$Y = \bar{c}_3\sinh(\lambda y) \tag{D.82}$$

但是，当 $\lambda = 0$ 时，式（D.70）成为 $Y'' = 0$，其解为直线方程

$$Y = c_3 + c_4 y \tag{D.83}$$

边界条件 $Y(0) = 0$ 意味着 $c_3 = 0$。因此，式（D.70）的解为

当 $n=0$ 时，$Y = c_{4_0}y$；并且，当 $n = 1, 2, 3, \cdots, \infty$ 时，$Y = \bar{c}_{3_n}\sinh\left(\dfrac{n\pi y}{a}\right)$

(D.84)

乘积 $X(x)Y(y)$ 的解为

当 $n=0$ 时，$u_0 = A_0 y$；当 $n = 1, 2, 3, \cdots, \infty$ 时，$u_n = A_n\sinh\left(\dfrac{n\pi y}{a}\right)\cos\left(\dfrac{n\pi x}{a}\right)$

(D.85)

应用叠加原理，建立所有这些解的线性组合：

$$u(x,y) = A_0 y + \sum_{n=1}^{\infty} A_n\sinh\left(\frac{n\pi y}{a}\right)\cos\left(\frac{n\pi x}{a}\right) \tag{D.86}$$

使用式（D.66）的最后一个边界条件，得到

$$u(x,b) = f(x) = A_0 b + \sum_{n=1}^{\infty} A_n\sinh\left(\frac{n\pi b}{a}\right)\cos\left(\frac{n\pi x}{a}\right) \tag{D.87}$$

为了求取 A_0 和 A_n 的值，将此式两边同乘以 $\cos(m\pi x/a)$，其中 m 为整数，即 $m = 1, 2, 3, \cdots, \infty$，再在区间 $(0, a)$ 上求积分，则

$$\int_0^a f(x)\cos\left(\frac{m\pi x}{a}\right)dx =$$

$$A_0 b\int_0^a \cos\left(\frac{m\pi x}{a}\right)dx + \sum_{n=1}^{\infty} A_n\sinh\left(\frac{n\pi b}{a}\right)\int_0^a \cos\left(\frac{m\pi x}{a}\right)\cos\left(\frac{n\pi x}{a}\right)dx \tag{D.88}$$

在区间 $(0, a)$ 上，当权重函数为 1 时余弦函数是正交函数，因此

$$\text{当 } n \neq m \text{ 时}, \int_0^a \cos\left(\frac{m\pi x}{a}\right)\cos\left(\frac{n\pi x}{a}\right)dx = 0 \tag{D.89}$$

但是，当 $n = m$ 时，应用积分表，如下所示，可以求得其积分值为 $(a/2)$

$$\int_0^a \cos^2\left(\frac{n\pi x}{a}\right)dx = \left[\frac{1}{2}x + \frac{1}{4\dfrac{n\pi}{a}}\sin\left(\frac{2n\pi x}{a}\right)\right]_0^a = \frac{a}{2} \tag{D.90}$$

利用式（D.89）和式（D.90），可以将式（D.88）简化为

$$\int_0^a f(x)\cos\left(\frac{n\pi x}{a}\right)\mathrm{d}x = A_0 b\int_0^a \cos\left(\frac{n\pi x}{a}\right)\mathrm{d}x + A_n \frac{a}{2}\sinh\left(\frac{n\pi b}{a}\right) \tag{D.91}$$

注意，因为 $n=m$ 时级数只剩一项，所以，此式中求和符号没有了。当 $n=0$ 时，$\cos(0)=1$ 且 $\sinh(0)=0$，因此式（D.91）可以进一步简化为

$$\int_0^a f(x)\mathrm{d}x = A_0 ab \tag{D.92}$$

于是，可求得 A_0 的值为

$$A_0 = \frac{1}{ab}\int_0^a f(x)\mathrm{d}x \tag{D.93}$$

当 $n>0$ 时，积分 $\int_0^a \cos(n\pi x/a)\mathrm{d}x$ 为零，由式（D.91）可得 A_n 的值为

$$A_n = \frac{2}{a\sinh\left(\frac{n\pi b}{a}\right)}\int_0^a f(x)\cos\left(\frac{n\pi x}{a}\right)\mathrm{d}x \tag{D.94}$$

这样，本命题的解由式（D.86）、以及式（D.93）和式（D.94）表示的 A_0 和 A_n 组成。要从 u 算回到 T，只要将式（D.64）变换为

$$T = T_0 u + T_0 \tag{D.95}$$

例 D.8　若 $f(x)=C$（常数），证明由式（D.87）给出的温度分布曲线在 y 方向上为直线。

解：

当 $f(x)=C$ 时，式（D.94）的积分为零，也就是 A_n 为零，即

$$A_n = \frac{2C}{a\sinh\left(\frac{n\pi b}{a}\right)}\int_0^a \cos\left(\frac{n\pi x}{a}\right)\mathrm{d}x = \frac{2C}{a\sinh\left(\frac{n\pi b}{a}\right)}\left[\frac{a}{n\pi}\sin\left(\frac{n\pi x}{a}\right)\right]_0^a = 0$$

式（D.93）的积分等于 Ca，所以 A_0 为 C/b：

$$A_0 = \frac{1}{ab}\int_0^a C\mathrm{d}x = \frac{Ca}{ab} = \frac{C}{b}$$

于是，式（D.86）简化为

$$u(x,y) = \frac{C}{b}y$$

将此式代入式（D.95），得到

$$T = T_0\frac{C}{b}y + T_0$$

这是 y 方向上的直线方程，此结果的物理含义是：平板的左右两边是理想的绝热体，而上下两边则保持常温。

D.3.3 周期函数和傅里叶级数

D3.1 节和 D3.2 节推导的解也可以利用傅里叶级数展开求得。

如果存在一个常数 $2p$，对于所有的 x 值，下式成立

$$f(x+2p)=f(x) \tag{D.96}$$

则函数 $f(x)$ 为周期函数。如果 $2p$ 是该等式成立的最小数值，$2p$ 就是函数的周期。众所周知，如下正弦函数和余弦函数是周期函数：

$$\left.\begin{aligned}\cos\left(\frac{n\pi(x+2p)}{p}\right)&=\cos\left(\frac{n\pi x}{p}\right)\\ \sin\left(\frac{n\pi(x+2p)}{p}\right)&=\sin\left(\frac{n\pi x}{p}\right)\end{aligned}\right\}\text{对于所有 } x \text{ 值} \tag{D.97}$$

任意一个周期为 $2p$ 的函数 $f(x)$ 都可以表示为如下级数的形式

$$\begin{aligned}f(x)=&\frac{a_0}{2}+a_1\cos\left(\frac{\pi x}{p}\right)+a_2\cos\left(\frac{2\pi x}{p}\right)+\cdots+a_n\cos\left(\frac{n\pi x}{p}\right)+\cdots\\&+b_1\sin\left(\frac{\pi x}{p}\right)+b_2\sin\left(\frac{2\pi x}{p}\right)+\cdots+b_n\sin\left(\frac{n\pi x}{p}\right)+\cdots\end{aligned} \tag{D.98}$$

其中

$$a_0=\frac{1}{p}\int_d^{d+2p}f(x)\mathrm{d}x \qquad (\text{对于所有 } d \text{ 值}) \tag{D.99}$$

$$a_n=\frac{1}{p}\int_d^{d+2p}f(x)\cos\left(\frac{n\pi x}{p}\right)\mathrm{d}x \qquad (\text{对于所有 } d \text{ 值}) \tag{D.100}$$

$$b_n=\frac{1}{p}\int_d^{d+2p}f(x)\sin\left(\frac{n\pi x}{p}\right)\mathrm{d}x \qquad (\text{对于所有 } d \text{ 值}) \tag{D.101}$$

以上 4 个方程构成了函数 $f(x)$ 的傅里叶级数展开式。

D.3.3.1 偶对称函数和奇对称函数

如果以下等式成立，则函数 $f(x)$ 为偶函数

$$f(-x)=f(x) \quad (\text{对于所有 } x)$$

如图 D.4a 所示，$f(x)$ 的曲线对称于垂直坐标轴。如果以下等式成立，则函数 $f(x)$ 为奇函数

$$f(-x)=-f(x) \quad (\text{对于所有 } x)$$

如图 D.4b 所示，其曲线对称于原点。在 $(-p, p)$ 区间上，偶函数的傅里叶级数展开式为

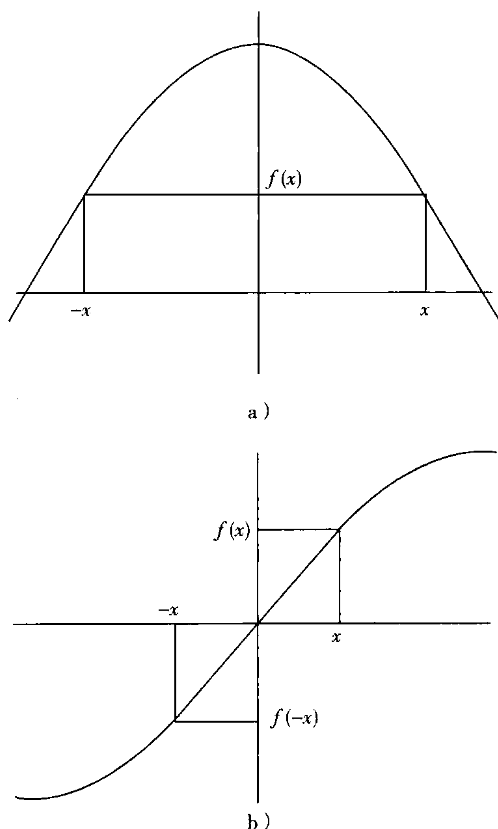


图　D.4
a）偶对称函数　b）奇对称函数

$$f(x)=\frac{a_0}{2}+\sum_{n=1}^{\infty}a_n\cos\left(\frac{n\pi x}{p}\right) \tag{D.102}$$

其中

$$a_0=\frac{2}{p}\int_0^p f(x)\,\mathrm{d}x \tag{D.103}$$

$$a_n=\frac{2}{p}\int_0^p f(x)\cos\left(\frac{n\pi x}{p}\right)\mathrm{d}x \tag{D.104}$$

在（$-p$，p）区间上，奇函数的傅里叶级数展开式为

$$f(x)=\sum_{n=1}^{\infty}b_n\sin\left(\frac{n\pi x}{p}\right) \tag{D.105}$$

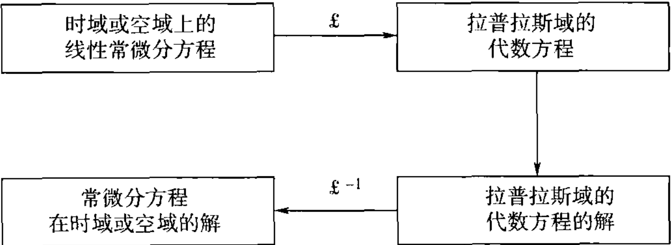
其中

$$b_n = \frac{2}{p}\int_0^p f(x)\sin\left(\frac{n\pi x}{p}\right)\mathrm{d}x \tag{D.106}$$

当$p=L$时，以上式（D.105）与D.3.1节的式（D.55）相同，因此，可以从式（D.106）得到常数A_n的值。同样，式（D.102）~式（D.104）3个等式也可以与D.3.2节的式（D.86）一起使用，以确定A_0和A_n的值。

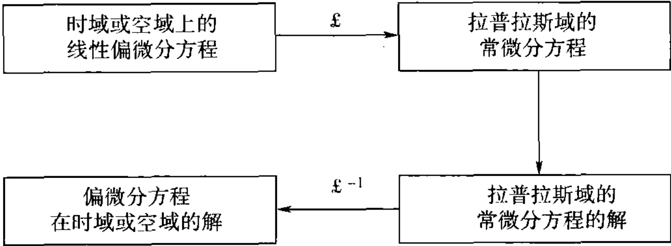
D.4 拉普拉斯变换

拉普拉斯变换是动态系统线性常微分方程和偏微分方程求解的一种很有用的方法。图D.5表示了利用拉普拉斯变换求解线性常微分方程的步骤，其中，符号£ 表示拉普拉斯变换运算，符号$£^{-1}$表示拉普拉斯反变换运算。D4.1节将给出这些运算的数学定义。对自变量为时域（或空间域）的常微分方程实施拉普拉斯变换，可以将其转化为拉普拉斯域（即s域）的代数方程，求解此代数方程，然后再应用拉普拉斯反变换，就可以求得原命题在时域上的解。



图D.5 利用拉普拉斯变换求解线性常微分方程的方法

图D.6显示了利用拉普拉斯变换求解线性偏微分方程的步骤，对自变量为时域或空间域的偏微分方程实施拉普拉斯变换之后，方程就被转化为拉普拉斯域的常微分方程，在拉普拉斯域应用常微分方程的解法求解此方程，然后再应用拉普拉斯反变换，就可以得到原偏微分方程在时域或空间域上的解。



图D.6 利用拉普拉斯变换求解线性偏微分方程的方法

D.4.1 拉普拉斯变换

拉普拉斯变换将函数$f(t)$ 从时域（或空间域）映射到 s 域，这里的 s 是定义在复平面上的变量，$s = \alpha + \beta i$。拉普拉斯变换定义为

$$£\ [f(t)] \equiv \bar{f}(s) = \int_0^{\infty} f(t)\mathrm{e}^{-st}\mathrm{d}t \tag{D.107}$$

此变换是一个广义积分，并不是对所有连续函数和所有 s 值都存在。对于函数 $f(t)$，拉普拉斯变换存在的充分条件是：

1）函数$f(t)$ 在每个$0<t<T$区间上都必须分段连续。所谓的函数在某个区间分段连续，就是指函数只有有限数目的有限断点。

2）函数$f(t)$ 在无穷大处必须是指数增长，所谓的函数f在$0<t<\infty$的无穷大处指数增长，就是指在足够大的 t 值下，存在常数 M 和 α，使得 $|f(t)| \leqslant M\mathrm{e}^{\alpha t}$ 。

3）对于任意有限值$\delta>0$，积分$\int_0^{\delta}|f(t)\,\mathrm{d}t|$ 必须存在，也就是其值有限。

拉普拉斯反变换将函数$\bar{f}(s)$ 从拉普拉斯域（即 s 域）映射回时域，即：

$$£^{-1}[\bar{f}(s)] = f(t) = \frac{1}{2\pi i}\int_{\gamma-i\infty}^{\gamma+i\infty}\bar{f}(s)\mathrm{e}^{st}\mathrm{d}s \tag{D.108}$$

求解此积分需要用到复变量、留数定理以及闭合线积分等相关知识。

下面我们推导部分函数的拉普拉斯变换，并将它们与另一些变换函数一起列于表 D.1，以便用于求解常微分方程和偏微分方程。通常并不直接从式（D.107）和式（D.108）的积分式计算拉普拉斯变换及其反变换，而是利用 Roberts 和 Kaufman 在 1966 年建立的详尽的拉普拉斯变换表来求变换以及反变换。MATLAB 等计算机软件也能计算拉普拉斯变换及其反变换。读者如果需要全面了解拉普拉斯变换的推导和相关论述，请参阅 Bequette（1998）和 Stephanopoulos（1984）等人的著作。

指数函数$f(t) = \mathrm{e}^{-at}$ 的拉普拉斯变换为

$$\begin{aligned}£\ [\mathrm{e}^{-at}] &= \int_0^{\infty}\mathrm{e}^{-at}\mathrm{e}^{-st}\mathrm{d}t = \int_0^{\infty}\mathrm{e}^{-(s+a)t}\mathrm{d}t \\ &= -\frac{1}{s+a}[\mathrm{e}^{-(s+a)t}]_0^{\infty} = \frac{1}{s+a}\end{aligned} \tag{D.109}$$

一阶导数 $\mathrm{d}f(t)/\mathrm{d}t$ 的拉普拉斯变换为

$$£\left[\frac{\mathrm{d}f(t)}{\mathrm{d}t}\right] = \int_0^{\infty}\frac{\mathrm{d}f(t)}{\mathrm{d}t}\mathrm{e}^{-st}\mathrm{d}t \tag{D.110}$$

利用部分积分（$\int u\mathrm{d}v = uv - \int v\mathrm{d}u$），可以求得

$$£\left[\frac{\mathrm{d}f(t)}{\mathrm{d}t}\right] = \int_0^{\infty}\frac{\mathrm{d}f(t)}{\mathrm{d}t}\mathrm{e}^{-st}\mathrm{d}t$$

$$
\begin{aligned}
&= [\mathrm{e}^{-st}f(t)]_0^{\infty} - \int_0^{\infty} f(t)(-s)\mathrm{e}^{-st}\mathrm{d}t \\
&= [0 - f(0)] + s\int_0^{\infty} f(t)\mathrm{e}^{-st}\mathrm{d}t \\
&= s\bar{f}(s) - f(0)
\end{aligned} \tag{D.111}
$$

n 阶导数 $\mathrm{d}^n f(t)/\mathrm{d}t^n$ 的拉普拉斯变换为

$$
\pounds\left[\frac{\mathrm{d}^n f(t)}{\mathrm{d}t^n}\right] = s^n\bar{f}(s) - s^{n-1}f(0) - s^{n-2}f'(0) - \cdots - sf^{(n-2)}(0) - f^{(n-1)}(0) \tag{D.112}
$$

求解此式需要 n 个初始条件，即 $f(0)$、$f'(0)$、$\cdots$、$f^{(n-1)}(0)$。

可以求得斜坡函 数 $f(t) = bt$ 的拉普拉斯变换为

$$
\begin{aligned}
\pounds[bt] &= \int_0^{\infty} bt\mathrm{e}^{-st}\mathrm{d}t \\
&= \left[-\frac{bt}{s}\mathrm{e}^{-st}\right]_0^{\infty} + \int_0^{\infty}\frac{b}{s}\mathrm{e}^{-st}\mathrm{d}t \\
&= (-0+0) + \frac{b}{s}\left[-\frac{1}{s}\mathrm{e}^{-st}\right]_0^{\infty} = \frac{b}{s^2}
\end{aligned} \tag{D.113}
$$

可以求得正弦函数的拉普拉斯变换为

$$
\begin{aligned}
\pounds[\sin(\omega t)] &= \int_0^{\infty}\sin(\omega t)\mathrm{e}^{-st}\mathrm{d}t \\
&= \int_0^{\infty}\frac{\mathrm{e}^{\mathrm{i}\omega t} - \mathrm{e}^{-\mathrm{i}\omega t}}{2\mathrm{i}}\mathrm{e}^{-st}\mathrm{d}t \\
&= \frac{1}{2i}\left[-\frac{\mathrm{e}^{-(s-\mathrm{i}\omega)t}}{s-\mathrm{i}\omega} + \frac{\mathrm{e}^{-(s+\mathrm{i}\omega)t}}{s+\mathrm{i}\omega}\right]_0^{\infty} \\
&= \frac{1}{2i}\left[\frac{1}{s-\mathrm{i}\omega} - \frac{1}{s+\mathrm{i}\omega}\right] = \frac{\omega}{s^2+\omega^2}
\end{aligned} \tag{D.114}
$$

同样，可以建立余弦函数的拉普拉斯变换，为

$$
\begin{aligned}
\pounds[\cos(\omega t)] &= \int_0^{\infty}\cos(\omega t)\mathrm{e}^{-st}\mathrm{d}t \\
&= \int_0^{\infty}\frac{\mathrm{e}^{\mathrm{i}\omega t} + \mathrm{e}^{-\mathrm{i}\omega t}}{2}\mathrm{e}^{-st}\mathrm{d}t \\
&= \frac{1}{2}\left[-\frac{\mathrm{e}^{-(s-\mathrm{i}\omega)t}}{s-\mathrm{i}\omega} - \frac{\mathrm{e}^{-(s+\mathrm{i}\omega)t}}{s+\mathrm{i}\omega}\right]_0^{\infty} \\
&= \frac{1}{2}\left[\frac{1}{s-\mathrm{i}\omega} + \frac{1}{s+\mathrm{i}\omega}\right] = \frac{s}{s^2+\omega^2}
\end{aligned} \tag{D.115}
$$

表 D.1　部分时域函数的拉普拉斯变换

$f(t)$	$\pounds\ [f(t)]$
单位阶跃函数：$S(t)=\begin{cases}0 & 当\ t<0\ 时\\ 1 & 当\ t>0\ 时\end{cases}$	$\frac{1}{s}$
常数：b	$\frac{b}{s}$
斜坡函数：bt	$\frac{b}{s^2}$
t^n	$\frac{n!}{s^{n+1}}$
e^{-at}	$\frac{1}{s+a}$
$t^n e^{-at}$	$\frac{n!}{(s+a)^{n+1}}$
$\frac{1}{a_1-a_2}(e^{-a_2t}-e^{-a_1t})$	$\frac{1}{(s+a_1)(s+a_2)}$
一阶导数：$\frac{df(t)}{dt}$	$s\bar{f}(s)-f(0)$
n 阶导数：$\frac{d^nf(t)}{dt^n}$	$s^n\bar{f}(s)-s^{n-1}f(0)-s^{n-2}f'(0)-\cdots$ $-sf^{(n-2)}(0)-f^{(n-1)}(0)$
$\sin(\omega t)$	$\frac{\omega}{s^2+\omega^2}$
$\cos(\omega t)$	$\frac{s}{s^2+\omega^2}$
$e^{-at}\sin(\omega t)$	$\frac{\omega}{(s+a)^2+\omega^2}$
$e^{-at}\cos(\omega t)$	$\frac{s+a}{(s+a)^2+\omega^2}$
$\int_0^t f(t)dt$	$\frac{1}{s}\bar{f}(s)$
$erfc\left(\frac{a^{\frac{1}{2}}}{2t^{\frac{1}{2}}}\right)$	$\frac{1}{s}e^{-a^{\frac{1}{2}}s^{\frac{1}{2}}}$

拉普拉斯变换有以下两个重要的定理，即初值定理和终值定理，这里不作详细证明。

初值定理：

$$\lim_{t\to 0}f(t)=\lim_{s\to\infty}[s\bar{f}(s)] \tag{D.116}$$

终值定理：

$$\lim_{t\to\infty} f(t) = \lim_{s\to 0}[s\bar{f}(s)] \tag{D.117}$$

其中

$$\bar{f}(s) = \int_0^{\infty} f(t)e^{-st}dt \quad \text{为} f(t) \text{的拉普拉斯变换。}$$

D.4.2　常微分方程的求解

考虑以下一阶非齐次常微分方程：

$$\frac{dy}{dt} + 3y = e^t \quad \text{且 } y(0) = 0 \tag{D.118}$$

对方程中的每一项求拉普拉斯变换，则

$$\pounds\left[\frac{dy}{dt}\right] + \pounds[3y] = \pounds[e^t]$$
$$s\bar{y}(s) - y(0) + 3\bar{y}(s) = \frac{1}{s-1} \tag{D.119}$$

利用初始条件 $y(0)=0$ 进行简化，并求 $\bar{y}(s)$，得

$$\bar{y}(s) = \frac{1}{(s+3)(s-1)} \tag{D.120}$$

此式的右边可以进行部分分式分解：

$$\frac{1}{(s+3)(s-1)} = \frac{c_1}{(s+3)} + \frac{c_2}{(s-1)} \tag{D.121}$$

为了计算常数 c_1 和 c_2 的值，先将此式乘以 $(s+3)$：

$$\frac{1}{(s-1)} = c_1 + \frac{c_2(s+3)}{(s-1)} \tag{D.122}$$

因为方程对于任意 s 值都必须成立，取 $s=-3$，将上式简化，并求 c_1 的值：

$$c_1 = \frac{1}{(s-1)} = \frac{1}{(-3-1)} = -\frac{1}{4} \tag{D.123}$$

用类似的方法，将式（D.121）乘以 $(s-1)$，取 $s=1$，求得 c_2 的值为

$$c_2 = \frac{1}{(s+3)} = \frac{1}{(1+3)} = \frac{1}{4} \tag{D.124}$$

将式（D.120）、式（D.121）、式（D.123）和式（D.124）这 4 个等式结合起来，可以构建如下 s 域的解：

$$\bar{y}(s) = \frac{-1/4}{(s+3)} + \frac{1/4}{(s-1)} \tag{D.125}$$

最后查表 D.1，求此式各项的拉普拉斯反变换，就可以得到微分方程的解：

$$\pounds^{-1}[\bar{y}(s)] = \pounds^{-1}\left[\frac{-1/4}{(s+3)}\right] + \pounds^{-1}\left[\frac{1/4}{(s-1)}\right]$$

$$y(t) = -\frac{1}{4}e^{-3t} + \frac{1}{4}e^{t} \tag{D.126}$$

建议读者用以下两种方法验证这个解的准确性：将解代入式（D.118）的原微分方程；用 D.1 节和 D.2 节介绍的方法求解该微分方程。

现在，我们再来看如何求解以下二阶非齐次常微分方程

$$\frac{d^2y}{dt^2} - 5\frac{dy}{dt} + 6y = \sin t \tag{D.127}$$

其初始条件为

$$y(0) = 0 \quad 且\left.\frac{dy}{dt}\right|_{t=0} = 0 \tag{D.128}$$

将此微分方程两边取拉普拉斯变换，得到：

$$\begin{aligned} &£\left[\frac{d^2y}{dt^2}\right] - £\left[5\frac{dy}{dt}\right] + £\left[6y\right] = £\left[\sin t\right] \\ &s^2\bar{y}(s) - sy(0) - y'(0) - 5(s\bar{y}(s) - y(0)) + 6\bar{y}(s) = \frac{1}{s^2+1} \end{aligned} \tag{D.129}$$

用式（D.128）的初始条件简化方程，并求 $\bar{y}(s)$：

$$\bar{y}(s) = \frac{1}{(s^2+1)(s^2-5s+6)} \tag{D.130}$$

注意，可以将 (s^2+1) 因式分解成 $(s+i)(s-i)$，将 (s^2-5s+6) 分解成 $(s-3)(s-2)$。于是，上式右边可以展开成部分分式：

$$\frac{1}{(s+i)(s-i)(s-3)(s-2)} = \frac{c_1}{(s+i)} + \frac{c_2}{(s-i)} + \frac{c_3}{(s-3)} + \frac{c_4}{(s-2)} \tag{D.131}$$

用本节前面所用的方法，可以求得这些常数的值为

$$c_1 = \frac{1}{10}\frac{1}{(1-i)}, \quad c_2 = \frac{1}{10}\frac{1}{(1+i)}, \quad c_3 = \frac{1}{10}, \quad c_4 = -\frac{1}{5} \tag{D.132}$$

将式（D.130）~式（D.132）3 个等式结合起来，得到 s 域的解为

$$\bar{y}(s) = \frac{\frac{1}{10}\frac{1}{(1-i)}}{(s+i)} + \frac{\frac{1}{10}\frac{1}{(1+i)}}{(s-i)} + \frac{\frac{1}{10}}{(s-3)} + \frac{-\frac{1}{5}}{(s-2)} \tag{D.133}$$

注意

$$\frac{1}{1-i}\frac{1+i}{1+i} = \frac{1+i}{2} \text{ 并且} \frac{1}{1+i}\frac{1-i}{1-i} = \frac{1-i}{2}$$

利用这两个恒等式，式（D.133）可以简化为

$$\bar{y}(s) = \frac{1}{10}\left(\frac{1+i}{2}\right)\frac{1}{(s+i)} + \frac{1}{10}\left(\frac{1-i}{2}\right)\frac{1}{(s-i)} + \frac{1}{10}\frac{1}{(s-3)} + \left(-\frac{1}{5}\right)\frac{1}{(s-2)} \tag{D.134}$$

此式的拉普拉斯反变换为

$$y(t) = \frac{1}{10}\left(\frac{1+\mathrm{i}}{2}\right)\mathrm{e}^{-\mathrm{i}t} + \frac{1}{10}\left(\frac{1-\mathrm{i}}{2}\right)\mathrm{e}^{\mathrm{i}t} + \frac{1}{10}\mathrm{e}^{3t} - \frac{1}{5}\mathrm{e}^{2t} \tag{D.135}$$

利用式（D.34）的欧拉恒等式，可以得到三角函数形式的解：

$$y(t) = \frac{1}{10}\cos t + \frac{1}{10}\sin t + \frac{1}{10}\mathrm{e}^{3t} - \frac{1}{5}\mathrm{e}^{2t} \tag{D.136}$$

用如下 MATLAB 指令可以验证这个解的正确性：

```
>> y = dsolve('D2y - 5 * Dy + 6 * y = sin(t)','y(0) =0','Dy(0) =0')
y =
1/10 * exp(3 * t) -1/5 * exp(2 * t) +1/10 * cos(t) +1/10 * sin(t)
```

作为练习，读者可以求一下（D.127）方程右边为 $\cos t$ 时的解。

D.4.3 偏微分方程的求解

将变量 u 替换为温度 T，方程（D.42）就变成了一维非稳态热传导问题的模型：

$$\frac{\partial T}{\partial t} = \alpha\frac{\partial^2 T}{\partial x^2} \tag{D.137}$$

该问题的初始条件和边界条件为

$$\begin{aligned}&\text{当 } t = 0 \text{ 时}, T(x,0) = T_0 \qquad \text{对于 } 0 \leqslant x \leqslant L\\ &\text{当 } t > 0 \text{ 时}, T(0,t) = T_0 \qquad \text{且 } T(L,t) = T_1\end{aligned} \tag{D.138}$$

为了便于求取该问题的解，我们用以下等式把变量 T 变换为变量 u：

$$u = \frac{T - T_0}{T_1 - T_0} \tag{D.139}$$

微分方程也就回到了式（D.42）

$$\frac{\partial u}{\partial t} = \alpha\frac{\partial^2 u}{\partial x^2} \tag{D.42}$$

初始条件和边界条件变成了如下齐次方程

$$\begin{aligned}&\text{当 } t = 0 \text{ 时}, u(x,0) = 0 \qquad \text{对于 } 0 \leqslant x \leqslant L\\ &\text{当 } t > 0 \text{ 时}, u(0,t) = 0 \qquad \text{且 } u(L,t) = 1\end{aligned} \tag{D.140}$$

将式（D.42）左边进行拉普拉斯变换，可得

$$\pounds\left[\frac{\partial u}{\partial t}\right] = s\bar{u} - u(x,0) \tag{D.141}$$

将式（D.42）右边进行拉普拉斯变换，可得

$$\pounds\left[\frac{\partial^2 u}{\partial x^2}\right] = \frac{\partial^2}{\partial x^2}(\pounds\ [u]) = \frac{\partial^2 \bar{u}}{\partial x^2} = \frac{\mathrm{d}^2 \bar{u}}{\mathrm{d}x^2} \tag{D.142}$$

这是因为 x 和 t 为自变量，并且，对于连续函数，拉氏变换与微分的次序可以互换。利用初始条件 $u(x,0)=0$，并结合式（D.141）、式（D.142）以及式（D.42），有

$$\alpha\frac{\mathrm{d}^2\bar{u}}{\mathrm{d}x^2}=s\bar{u} \tag{D.143}$$

再将边界条件变换到 s 域：

$$\text{当}\ x=0\ \text{时},u(0,t)=0,\quad \pounds\left[u(0,t)\right]=\bar{u}(0,s)=0$$

$$\text{当}\ x=L\ \text{时},u(L,t)=1,\quad \pounds\left[u(L,t)\right]=\bar{u}(L,s)=\frac{1}{s} \tag{D.144}$$

式（D.143）是 $\bar{u}$ 的二阶常微分方程，用 D.2 节的方法可以求解此方程，得到解：

$$\bar{u}=c_1\mathrm{e}^{\sqrt{\frac{s}{\alpha}}x}+c_2\mathrm{e}^{-\sqrt{\frac{s}{\alpha}}x} \tag{D.145}$$

利用边界条件确定常数 c_1 和 c_2 的值：

$$c_1=\frac{1}{s}\frac{1}{\left(\mathrm{e}^{\sqrt{\frac{s}{\alpha}}L}-\mathrm{e}^{-\sqrt{\frac{s}{\alpha}}L}\right)},\qquad c_2=-\frac{1}{s}\frac{1}{\left(\mathrm{e}^{\sqrt{\frac{s}{\alpha}}L}-\mathrm{e}^{-\sqrt{\frac{s}{\alpha}}L}\right)} \tag{D.146}$$

代入这两个常数的值，式（D.145）变为

$$\bar{u}=\frac{1}{s}\frac{\left(\mathrm{e}^{\sqrt{\frac{s}{\alpha}}x}-\mathrm{e}^{-\sqrt{\frac{s}{\alpha}}x}\right)}{\left(\mathrm{e}^{\sqrt{\frac{s}{\alpha}}L}-\mathrm{e}^{-\sqrt{\frac{s}{\alpha}}L}\right)} \tag{D.147}$$

此式右边可以展开成部分分式：

$$\frac{1}{s}\frac{\left(\mathrm{e}^{\sqrt{\frac{s}{\alpha}}x}-\mathrm{e}^{-\sqrt{\frac{s}{\alpha}}x}\right)}{\left(\mathrm{e}^{\sqrt{\frac{s}{\alpha}}L}-\mathrm{e}^{-\sqrt{\frac{s}{\alpha}}L}\right)}=\frac{A}{s}+\frac{B}{\left(\mathrm{e}^{\sqrt{\frac{s}{\alpha}}L}-\mathrm{e}^{-\sqrt{\frac{s}{\alpha}}L}\right)} \tag{D.148}$$

为了确定 A 的值，将上式两边同乘以 s，并设 $s=0$，有：

$$A=\frac{\left(\mathrm{e}^{\sqrt{\frac{s}{\alpha}}x}-\mathrm{e}^{-\sqrt{\frac{s}{\alpha}}x}\right)}{\left(\mathrm{e}^{\sqrt{\frac{s}{\alpha}}L}-\mathrm{e}^{-\sqrt{\frac{s}{\alpha}}L}\right)} \tag{D.149}$$

但是，由于 $s=0$，不能直接由此式求得 A 值。所以，必须用如下洛必达（L'Hospital）法则求 A 值：

$$A=\lim_{s\to0}\frac{\left(\mathrm{e}^{\sqrt{\frac{s}{\alpha}}x}-\mathrm{e}^{-\sqrt{\frac{s}{\alpha}}x}\right)}{\left(\mathrm{e}^{\sqrt{\frac{s}{\alpha}}L}-\mathrm{e}^{-\sqrt{\frac{s}{\alpha}}L}\right)}=\frac{x}{L} \tag{D.150}$$

为了确定 B 的值，将式（D.148）两边同乘以 $\left(\mathrm{e}^{\sqrt{\frac{s}{\alpha}}L}-\mathrm{e}^{-\sqrt{\frac{s}{\alpha}}L}\right)$，并设其值为 0，有：

$$B=\frac{1}{s}\left(\mathrm{e}^{\sqrt{\frac{s}{\alpha}}x}-\mathrm{e}^{-\sqrt{\frac{s}{\alpha}}x}\right) \tag{D.151}$$

由三角恒等式可得

$$\left(e^{\sqrt{\frac{s}{\alpha}}L} - e^{-\sqrt{\frac{s}{\alpha}}L}\right) = 2\sinh\left(\sqrt{\frac{s}{\alpha}}L\right) \tag{D.152}$$

要使该恒等式等于0，以下等式必须成立：

$$\sqrt{\frac{s}{\alpha}}L = in\pi \qquad 或 \qquad s_n = -\frac{\alpha n^2\pi^2}{L^2}, n = 1,2,\cdots,\infty \tag{D.153}$$

利用式（D.148）~式（D.152）这几个等式，可以构建出如下式（D.147）的$\bar{u}$：

$$\bar{u} = \frac{1}{s}\frac{x}{L} + \frac{1}{s}\frac{\left(e^{\sqrt{\frac{s}{\alpha}}x} - e^{-\sqrt{\frac{s}{\alpha}}x}\right)}{2\sinh\left(\sqrt{\frac{s}{\alpha}}L\right)} \tag{D.154}$$

求上式的拉普拉斯反变换：

$$u = £^{-1}[\bar{u}] = £^{-1}\left[\frac{1}{s}\frac{x}{L}\right] + £^{-1}\left[\frac{1}{s}\frac{\left(e^{\sqrt{\frac{s}{\alpha}}x} - e^{-\sqrt{\frac{s}{\alpha}}x}\right)}{2\sinh\left(\sqrt{\frac{s}{\alpha}}L\right)}\right] \tag{D.155}$$

查 Roberts 和 Kaufman1966 年出版的拉普拉斯变换表（第284页），可得解：

$$u = \frac{x}{L} + \sum_{n=1}^{\infty}\frac{2}{n\pi}(-1)^n\sin\left(\frac{n\pi x}{L}\right)e^{-\frac{\alpha n^2\pi^2 t}{L}} \tag{D.156}$$

最后，将式（D.139）代入，把变量u变换回变量T：

$$T = u(T_1 - T_0) + T_0 \tag{D.157}$$

例D.9 免疫血细胞（即白细胞）在人造血管材料中的迁移机制是材料的生物相容性的重要问题。不考虑对流，迁移机制的最简单形式可以用如下扩散方程来模拟：

$$\frac{\partial c}{\partial t} = D\frac{\partial^2 c}{\partial x^2}$$

初始时，人工材料中的白细胞浓度为0，即

当$t = 0$时，对于所有x，$c(x,0) = 0$

设$t > 0$时，材料表面的白细胞浓度为c_A，即

当$t > 0$时，在$x = 0$处，$c(0,t) = c_A$

对于这个边界条件，假设相对于材料的厚度，白细胞扩散渗入材料的距离非常小，即

在$x = \infty$处，对于所有t，有$c(\infty,t) = 0$

请利用这些初始条件和边界条件，求解微分方程并作图显示白细胞的迁移曲线。

解：

为了简化方程的解析解，先将浓度变量重新定义为

$$u = \frac{c}{c_A}$$

则微分方程变为

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2}$$

初始条件和边界条件变成了如下齐次方程：

$$\text{当 } t = 0 \text{ 时}, u(x,0) = 0 \quad \text{对于所有 } x$$
$$\text{当 } x = 0 \text{ 时}, u(0,t) = 1 \quad \text{对于 } t > 0$$
$$\text{当 } x = \infty \text{ 时}, u(\infty,t) = 0 \quad \text{对于所有 } t$$

求扩散方程的拉普拉斯变换，并利用初始条件，可得

$$D\frac{\mathrm{d}^2\bar{u}}{\mathrm{d}x^2} = s\bar{u}$$

此结果与式（D.143）完全相同。将边界条件进行拉普拉斯变换，可得

$$\text{当 } x = 0 \text{ 时}, u(0,t) = 1, \quad \pounds\,[u(0,t)] = \bar{u}(0,s) = \frac{1}{s}$$

$$\text{当 } x = \infty \text{ 时}, u(\infty,t) = 0, \pounds\,[u(\infty,t)] = \bar{u}(\infty,s) = 0$$

拉普拉斯变换之后的微分方程的解就是

$$\bar{u} = c_1 e^{\sqrt{\frac{s}{D}}x} + c_2 e^{-\sqrt{\frac{s}{D}}x}$$

由 $x = \infty$ 处的边界条件可知积分常数 c_1 为 0。由 $x = 0$ 处的边界条件求得积分常数 c_2 的值为

$$c_2 = \frac{1}{s}$$

于是，微分方程的解成为

$$\bar{u} = \frac{1}{s}e^{-\sqrt{\frac{s}{D}}x}$$

表 D.1 中有该式的拉普拉斯反变换，为

$$u = \operatorname{erfc}\left(\frac{x}{2\sqrt{Dt}}\right) = 1 - \operatorname{erf}\left(\frac{x}{2\sqrt{Dt}}\right)$$

erf( ) 是误差函数，其定义是

$$\operatorname{erf}(\theta) = \frac{2}{\sqrt{\pi}}\int_0^{\theta} e^{-t^2}\mathrm{d}t$$

最后，用归一化浓度表示的解为

$$\frac{c}{c_A} = 1 - \operatorname{erf}\left(\frac{x}{2\sqrt{Dt}}\right)$$

下面是该微分方程求解的 MATLAB 程序，图 D.7 为程序运行的输出结果：

```
% Plotting the migration profiles of the immune blood cells
clear; clc
D=0.01;
x=[0:0.01:1];
nc=0;
for t=0:0.5:5
      nc=nc+1;
      c(nc,:)=(1-erf(x/(2*sqrt(D*t))));
end
plot(x,c)
title('Migration profiles of immune blood cells')
xlabel('Position, x');
ylabel('Normalized Concentration')
text(0.4, 0.4,'Time');
```

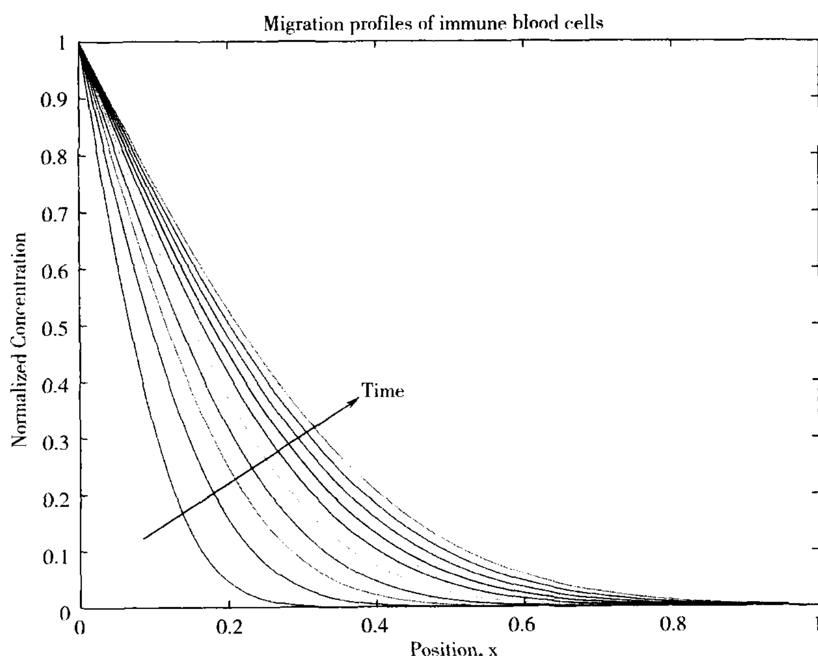


图 D.7 白细胞迁移的分布曲线

有关利用拉普拉斯变换求解偏微分方程的内容就介绍这些。

D.5 参考文献

Bequette, B. W. 1998. *Process Dynamics: Modeling, Analysis, and Simulation*. Upper Saddle River, NJ; Prentice Hall, Inc.

Boyce, W. E. and DiPrima, R. C. 2001. *Elementary Differential Equations*. New York: John Wiley & Sons, Inc.

O'Neil, P. V. 2003. *Advanced Engineering Mathematics*, 5th ed. Pacific Grove, CA: Brooks/Cole-Thompson Learning, Inc.

Roberts, G.E and Kaufman, H. 1966. *Table of Laplace Transforms*. Philadelphia, PA: W. B. Saunders Company.

Stephanopoulos, G. 1984. *Chemical Process Control: An Introduction to Theory and Practice*. Upper Saddle River, NJ; Prentice Hall, Inc.

Zill, D. G. and Cullen, M. R. 2000. *Advanced Engineering Mathematics*. Boston, MA: James & Bartlett Publishers.

附录 E　数值稳定性等问题

微分方程的数值积分有 3 个重要的问题，即：误差传播、稳定性以及解的收敛性。常微分方程的解需要考虑两种稳定性：解析解自身的稳定性和数值解的稳定性。如同 7.6 节所述，解析解自身的稳定性决定于命题的数学公式，它与微分方程雅可比矩阵的特征值有关。而数值解的稳定性与数值积分算法的误差传播有关，而误差传播的大小则取决于求数值解的差分方程的特征根。在本附录中，我们将系统地考察几种数值积分算法的误差传播及其稳定性，并说明通过合理选择步长和积分算法来减小误差的方法。

E.1　欧拉法的稳定性

考虑如下线性形式的初值问题微分方程：

$$\frac{\mathrm{d}y}{\mathrm{d}t} = \lambda y \tag{E.1}$$

其初始条件为

$$y(t_0) = y_0 \tag{E.2}$$

并进一步假设 λ 为实数，且 y_0 为有限值，则此微分方程的解析解为

$$y(t) = y_0 \mathrm{e}^{\lambda t} \tag{E.3}$$

当 $\lambda < 0$ 时，这个解本身是稳定的，也就是

$$\lim_{t \to \infty} y(t) = 0 \tag{E.4}$$

现在，考察其数值解的稳定性，假设用显式欧拉法求其数值解，并且暂时忽略截断误差和舍入误差。利用式（7.25）可得如下递推方程

$$y_{n+1} = y_n + h\lambda y_n \tag{E.5}$$

重排之后，得到一阶齐次差分方程

$$y_{n+1} - (1 + h\lambda) y_n = 0 \tag{E.6}$$

应用移位算子的定义 $Ey_n = y_{n+1}$，可得

$$Ey_n - (1 + h\lambda) y_n = 0 \tag{E.7}$$

产生特征方程

$$E - (1 + h\lambda) = 0 \tag{E.8}$$

其特征根为

$$\mu_1 = (1 + h\lambda) \tag{E.9}$$

由此得到差分方程（E.6）的解为

$$y_n = C(1 + h\lambda)^n \tag{E.10}$$

由 $t = t_0$ 时的初始条件计算常数 C：

$$n = 0 \qquad y_n = y_0 = C \tag{E.11}$$

于是，解的最终形式为

$$y_n = y_0(1 + h\lambda)^n \tag{E.12}$$

原微分方程是一个初值问题，因此 n 可以无限增加。由于解 y_n 是 $(1 + h\lambda)^n$ 的函数，所以解的性质由 $(1 + h\lambda)$ 的值决定。

如果下式成立，则数值解是绝对稳定的

$$\lim_{n\to\infty} y_n = 0 \tag{E.13}$$

如果

$$|1 + h\lambda| \leqslant 1 \tag{E.14}$$

那么，用显式欧拉法得到的微分方程（E.1）的数值解是绝对稳定的。由于 $(1 + h\lambda)$ 是特征方程（E.8）的特征根，因此，当多步数值方法存在多个特征根时，绝对稳定的另一种定义是

$$|\mu_i| \leqslant 1 \qquad i = 1,2,\cdots,k \tag{E.15}$$

不等式（E.14）等价于

$$-2 \leqslant h\lambda \leqslant 0 \tag{E.16}$$

此式给出了如下稳定解的积分步长 h 的极限值：由于 h 为正，因此 $\lambda < 0$，并且

$$h \leqslant \frac{2}{|\lambda|} \tag{E.17}$$

这个不等式是有限的总体稳定边界，显式欧拉法称为条件稳定的。任何具有无限总体稳定边界的方法称为无条件稳定的。

为了简化推导过程，我们在本节开始处假设 λ 为实数，其实没有必要，λ 也可以是复数。如果存在复数根 $(\alpha \pm \beta \mathrm{i})$，并且其模 $(r = \sqrt{\alpha^2 + \beta^2})$ 小于等于1，即

$$|r| \leqslant 1 \tag{E.18}$$

那么，解就是稳定的，是一个收敛的阻尼振荡。（E.16）和（E.18）两个不等式表示的是复平面上半径为1的圆。图E.1为欧拉法和龙格-库塔法的数值解的稳定区域，位于圆圈内的所有 $h\lambda$ 值产生欧拉法（等价于一阶龙格-库塔积分法）求解（E.1）方程的稳定数值解。

现在，我们再来考虑欧拉法的截断误差和舍入误差，将使用差分方程说明这两种误差在数值解中的传播。首先，从如下初值问题的非线性形式开始：

$$\frac{\mathrm{d}y}{\mathrm{d}t} = f(t,y) \tag{E.19}$$

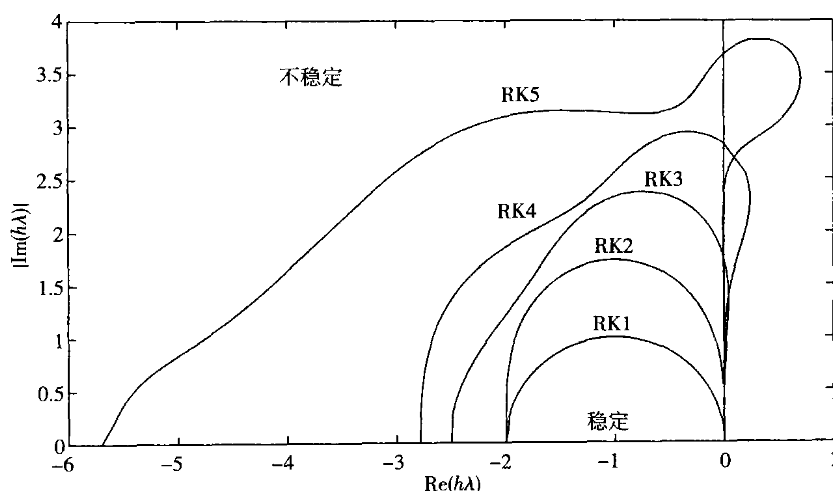


图 E.1 一至五阶龙格-库塔法在复平面上的稳定区域
(其中一阶龙格-库塔法也就是显式欧拉法)

其初始条件为

$$y(t_0) = y_0 \tag{E.20}$$

定义第 $(n+1)$ 步时数值解的累积误差为

$$\varepsilon_{n+1} = y_{n+1} - y(t_{n+1}) \tag{E.21}$$

式中 $y(t_{n+1})$ ——y 的准确值;

y_{n+1} ——y 在 t_{n+1} 处的计算值。

这里，我们把准确解 $y(t_{n+1})$ 表示成泰勒级数展开式，并且只写出欧拉法所需的级数项，即:

$$y(t_{n+1}) = y(t_n) + hf(t_n, y(t_n)) + T_{E,n+1} \tag{E.22}$$

式中 $T_{E,n+1}$ ——第 $(n+1)$ 步时的局部截断误差。

再写出由显示欧拉公式计算得到的 y_{n+1} 值:

$$y_{n+1} = y_n + hf(t_n, y_n) + R_{E,n+1} \tag{E.23}$$

式中 $R_{E,n+1}$ ——计算机在第 $(n+1)$ 步引入的舍入误差。

从式 (E.21) ~式 (E.23) 可得:

$$\varepsilon_{n+1} = y_n - y(t_n) + h[f(t_n, y_n) - f(t_n, y(t_n))] - T_{E,n+1} + R_{E,n+1} \tag{E.24}$$

简化为

$$\varepsilon_{n+1} = \varepsilon_n + h[f(t_n, y_n) - f(t_n, y(t_n))] - T_{E,n+1} + R_{E,n+1} \tag{E.25}$$

利用如下中值定理

$$f(t_n, y_n) - f(t_n, y(t_n)) = \left.\frac{\partial f}{\partial y}\right|_{\alpha, x_n} [y_n - y(t_n)] \quad y_n < \alpha < y(t_n) \tag{E.26}$$

可以将式（E. 25）简化为

$$\varepsilon_{n+1} - \left[1 + h\left.\frac{\partial f}{\partial y}\right|_{\alpha, x_n}\right]\varepsilon_n = -T_{E,n+1} + R_{E,n+1} \tag{E.27}$$

这是一个一阶变系数非齐次差分方程，只能用迭代法求解。但是，如果有以下假定：

$$T_{E,n+1} = T_E = 常数 \tag{E.28}$$

$$R_{E,n+1} = R_E = 常数 \tag{E.29}$$

$$\left.\frac{\partial f}{\partial y}\right|_{\alpha, x_n} = \lambda = 常数 \tag{E.30}$$

则式（E. 27）就可以简化为

$$\varepsilon_{n+1} - (1 + h\lambda)\varepsilon_n = -T_E + R_E \tag{E.31}$$

该方程的解由齐次方程的解加上特解构成，即

$$\varepsilon_n = C_1(1 + h\lambda)^n + \frac{-T_E + R_E}{1 - (1 + h\lambda)} \tag{E.32}$$

对比式（E. 6）和式（E. 31）这两个方程，可见解 y_n 的特征方程与误差 ε_n 的特征方程相同。特解则由式（E. 31）中的截断误差和舍入误差这两项决定。假设微分方程的初始条件为 $\varepsilon_0 = 0$，也就是没有误差，据此可求得常数 C_1 的值，于是，描述传播误差大小的方程最终成为

$$\varepsilon_n = \frac{-T_E + R_E}{h\lambda}[(1 + h\lambda)^n - 1] \tag{E.33}$$

仔细分析这个方程可以得到许多信息。正如预期的那样，$(1 + h\lambda)$ 的值是传播误差大小的决定因素，先看 h 为有限固定步长时的情况，当积分步数增加，n 值很大时，即 $n \to \infty$ 时的误差极值为

$$\lim_{n\to\infty} |\varepsilon_n| = \frac{-T_E + R_E}{h\lambda} \qquad 当 |1 + h\lambda| < 1 时 \tag{E.34}$$

$$\lim_{n\to\infty} |\varepsilon_n| = \infty \qquad 当 |1 + h\lambda| > 1 时 \tag{E.35}$$

在以上第一种情况下，$\lambda < 0, 0 < h < 2/|\lambda|$，此时误差有限，数值解稳定。数值解与准确解的差别只是一个有限数值 $(-T_E + R_E)/h\lambda$，它是截断误差、舍入误差、步长以及微分方程特征值的函数。

在第二种情况下，$\lambda > 0$，$h > 0$，此时误差无限大，数值解不稳定。如果 $\lambda > 0$，准确解本身就是不稳定的，因此我们引入相对误差的概念，其定义是

$$相对误差 = \frac{\varepsilon_n}{y_n} \tag{E.36}$$

将式（E.12）和式（E.33）代入，得到相对误差为

$$\frac{\varepsilon_n}{y_n} = \frac{-T_E + R_E}{y_0 h\lambda}\left[1 - \frac{1}{(1+h\lambda)^n}\right] \tag{E.37}$$

当 $\lambda > 0$ 时，此相对误差有限；当 $\lambda < 0$ 时，则无限。于是，我们可以得出结论：对于本身稳定的微分方程，绝对误差是数值计算稳定性的合适的评判指标，而对于本身不稳定的微分方程，则相对误差才是正确的判断指标。

如果积分区间固定，即 $0 < t < \alpha$，则

$$h = \frac{\alpha}{n} \tag{E.38}$$

若增加积分步数，使 n 值极大，于是导致 $h \to 0$。

如果有

$$\lim_{h\to 0} |\varepsilon_n| = 0 \tag{E.39}$$

则此数值方法称为收敛的。如果没有舍入误差，欧拉法等大多数积分方法都是收敛的，因为

$$\lim_{h\to 0} T_E = 0 \tag{E.40}$$

这时，式（E.39）成立。但是，数值计算实际上不可能没有舍入误差。

当 $h \to 0$ 时，截断误差趋于0，舍入误差就成为误差传播的主要因素：

$$\lim_{h\to 0} |\varepsilon_n| = R_E \lim_{h\to 0} \frac{(1+h\lambda)^n - 1}{h\lambda} \tag{E.41}$$

根据洛必达（L'Hospital）法则，当积分步数极大时，舍入误差无限传播，即

$$\lim_{h\to 0} \varepsilon_n = R_E[\infty] \tag{E.42}$$

这是数值方法中令人左右为难的问题，减小积分步长可以降低截断误差，但是，所需要的计算量增加，从而增加了舍入误差。

同理，可以分析隐式欧拉法（即向后欧拉法），其解为

$$y_{n+1} = \frac{y_0}{(1-h\lambda)^n} \tag{E.43}$$

传播误差为

$$\varepsilon_{n+1} = \frac{-T_E + R_E}{h\lambda}(1-h\lambda)\left[\frac{1}{(1-h\lambda)^n} - 1\right] \tag{E.44}$$

当 $\lambda < 0$ 且 $0 < h < \infty$ 时，解是稳定的，即

$$\lim_{n\to\infty} y_n = 0 \tag{E.45}$$

并且误差有限：

$$\lim_{n\to\infty}\varepsilon_n = -\frac{-T_E + R_E}{h\lambda}(1 - h\lambda) \tag{E.46}$$

这里的步长大小没有限制，因此，当 $\lambda<0$ 时，隐式欧拉法是绝对稳定的。另一方面，当 $\lambda>0$ 时，只有当以下不等式成立时才有稳定解

$$|1 - h\lambda| \leqslant 1 \tag{E.47}$$

这就使步长大小受到了限制：

$$-2 \leqslant h\lambda < 0 \tag{E.48}$$

结论是隐式欧拉法稳定范围大于显式欧拉法（参见表 E. 1）。

E. 2　龙格-库塔法的稳定性

应用与上一节类似的方法，可以推导出龙格-库塔法的递推公式及其相应的特征根（Lapidus 和 Sienfeld，1971）。对于（E. 1）的微分方程，有：
二阶龙格-库塔法：

$$y_{n+1} = \left(1 + h\lambda + \frac{1}{2}h^2\lambda^2\right)y_n \tag{E.49}$$

$$\mu_1 = 1 + h\lambda + \frac{1}{2}h^2\lambda^2 \tag{E.50}$$

三阶龙格-库塔法：

$$y_{n+1} = \left(1 + h\lambda + \frac{1}{2}h^2\lambda^2 + \frac{1}{6}h^3\lambda^3\right)y_n \tag{E.51}$$

$$\mu_1 = 1 + h\lambda + \frac{1}{2}h^2\lambda^2 + \frac{1}{6}h^3\lambda^3 \tag{E.52}$$

四阶龙格-库塔法：

$$y_{n+1} = \left(1 + h\lambda + \frac{1}{2}h^2\lambda^2 + \frac{1}{6}h^3\lambda^3 + \frac{1}{24}h^4\lambda^4\right)y_n \tag{E.53}$$

$$\mu_1 = 1 + h\lambda + \frac{1}{2}h^2\lambda^2 + \frac{1}{6}h^3\lambda^3 + \frac{1}{24}h^4\lambda^4 \tag{E.54}$$

五阶龙格-库塔法：

$$y_{n+1} = \left(1 + h\lambda + \frac{1}{2}h^2\lambda^2 + \frac{1}{6}h^3\lambda^3 + \frac{1}{24}h^4\lambda^4 + \frac{1}{120}h^5\lambda^5 + \frac{0.5625}{720}h^6\lambda^6\right)y_n \tag{E.55}$$

$$\mu_1 = 1 + h\lambda + \frac{1}{2}h^2\lambda^2 + \frac{1}{6}h^3\lambda^3 + \frac{1}{24}h^4\lambda^4 + \frac{1}{120}h^5\lambda^5 + \frac{0.5625}{720}h^6\lambda^6 \tag{E.56}$$

式（E.55）和式（E.56）两式右边最后一项是五阶龙格-库塔法特有的，出自 Constantinides 和 Mostoufi（1999）著作的表5.2。对于不同的五阶公式，这一项有所不同。

绝对稳定的条件是

$$|\mu_i| \leqslant 1 \qquad i = 1,2,\cdots,k \tag{E.57}$$

将其用于以上这些算法，所得到的实数绝对稳定界限列于表E.1，复平面上的稳定区域见图E.2。通常，阶数越高，稳定的范围也越大。

表 E.1 实数稳定界限

算 法	稳定界限
显示欧拉法	$-2 \leqslant h\lambda < 0$
隐式欧拉法	$\begin{cases} 0 < h < \infty & 当\ \lambda < 0 \\ -2 \leqslant h\lambda < 0 & 当\ \lambda > 0 \end{cases}$
改进欧拉法（预测-校正法）	$-1.077 \leqslant h\lambda < 0$
二阶龙格-库塔法	$-2 \leqslant h\lambda < 0$
三阶龙格-库塔法	$-2.5 \leqslant h\lambda < 0$
四阶龙格-库塔法	$-2.785 \leqslant h\lambda < 0$
五阶龙格-库塔法	$-5.7 \leqslant h\lambda < 0$
亚当斯法	$-0.546 \leqslant h\lambda < 0$
亚当斯-莫尔顿法	$-1.285 \leqslant h\lambda < 0$

E.3 多步算法的稳定性

应用与第一节类似的方法，可以推导出改进欧拉法（Modified Euler）、亚当斯法（Adams）和亚当斯-莫尔顿法（Adams-Moulton）的递推公式及其相应的特征根（Lapidus 和 Sienfeld，1971）。对于式（E.1）的微分方程，有

改进欧拉法（预测和校正相结合）：

$$y_{n+1} = (1 + h\lambda + h^2\lambda^2)y_n \tag{E.58}$$

$$\mu_1 = 1 + h\lambda + h^2\lambda^2 \tag{E.59}$$

亚当斯法：

$$y_{n+1} = \left(1 + \frac{23}{12}h\lambda\right)y_n - \left(\frac{4}{3}h\lambda\right)y_{n-1} + \left(\frac{5}{12}h\lambda\right)y_{n-2} \tag{E.60}$$

$$\mu^3 - \left(1 + \frac{23}{12}h\lambda\right)\mu^2 + \left(\frac{4}{3}h\lambda\right)\mu - \left(\frac{5}{12}h\lambda\right) = 0 \tag{E.61}$$

亚当斯-莫尔顿法（预测和校正相结合）：

$$y_{n+1} = \left(1 + \frac{7}{6}h\lambda + \frac{55}{64}h^2\lambda^2\right)y_n - \left(\frac{5}{24}h\lambda + \frac{59}{64}h^2\lambda^2\right)y_{n-1}$$
$$+ \left(\frac{1}{24}h\lambda + \frac{37}{64}h^2\lambda^2\right)y_{n-2} - \left(\frac{9}{64}h^2\lambda^2\right)y_{n-3} \quad (\text{E}.62)$$

$$\mu^4 - \left(1 + \frac{7}{6}h\lambda + \frac{55}{64}h^2\lambda^2\right)\mu^3 + \left(\frac{5}{24}h\lambda + \frac{59}{64}h^2\lambda^2\right)\mu^2$$
$$- \left(\frac{1}{24}h\lambda + \frac{37}{64}h^2\lambda^2\right)\mu + \left(\frac{9}{64}h^2\lambda^2\right) = 0 \quad (\text{E}.63)$$

绝对稳定的条件是

$$|\mu_i| \leqslant 1 \qquad i = 1,2,\cdots,k \quad (\text{E}.57)$$

将其用于以上这些算法，求得的实数绝对稳定界限列于表 E. 1，复平面上的稳定区域如图 E. 2 所示。

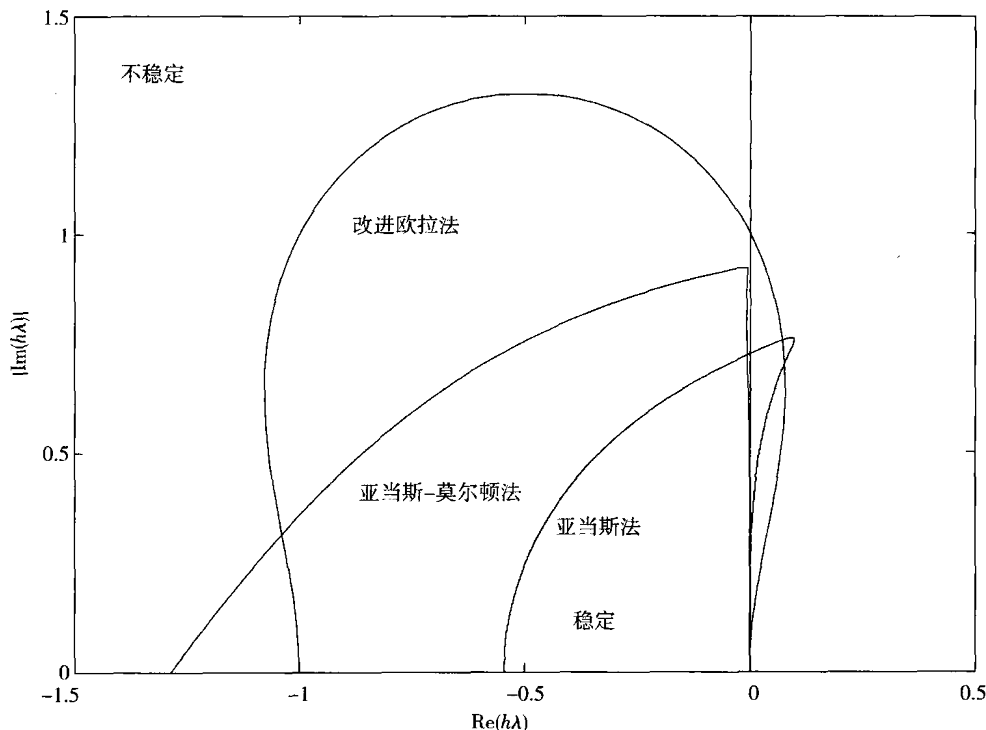


图 E. 2 改进欧拉法（即欧拉预测-校正法）、亚当斯法和亚当斯-莫尔顿法在复平面上的稳定区域

E.4 偏微分方程数值方法的稳定性

本节将利用著名的冯·诺依曼方法讨论偏微分方程有限差分近似解的稳定性。所采用的方法是先引进一个用有限傅里叶级数表示的初始误差，然后考察该误差在数值求解过程中的传播情况。由于冯·诺依曼方法是用于初值问题的，因此，在此将其用于分析 8.5.2 节导出的抛物型方程和 8.5.3 节导出的双曲型方程的显式方法的稳定性。

将第 (m,n) 步计算时的误差 $\varepsilon_{m,n}$ 定义为有限差分近似解 $u_{m,n}$ 与微分方程准确解 $\bar{u}_{m,n}$ 之间的差，即

$$\varepsilon_{m,n} \equiv u_{m,n} - \bar{u}_{m,n} \tag{E.64}$$

根据式（8.22）抛物型偏微分方程的显式有限差分解式（8.61），可以写出如下 $u_{m,n+1}$ 和 $\bar{u}_{m,n+1}$ 的计算式：

$$u_{m,n+1} = \left(\frac{\alpha\Delta t}{\Delta x^2}\right)u_{m+1,n} + \left(1 - 2\frac{\alpha\Delta t}{\Delta x^2}\right)u_{m,n} + \left(\frac{\alpha\Delta t}{\Delta x^2}\right)u_{m-1,n} + R_{E_{m,n+1}} \tag{E.65}$$

$$\bar{u}_{m,n+1} = \left(\frac{\alpha\Delta t}{\Delta x^2}\right)\bar{u}_{m+1,n} + \left(1 - 2\frac{\alpha\Delta t}{\Delta x^2}\right)\bar{u}_{m,n} + \left(\frac{\alpha\Delta t}{\Delta x^2}\right)\bar{u}_{m-1,n} + T_{E_{m,n+1}} \tag{E.66}$$

式中　$R_{E_{m,n+1}}$，$T_{E_{m,n+1}}$——第 $(m,n+1)$ 步的舍入误差和截断误差。

由式（E.64）~式（E.66）可得

$$\varepsilon_{m,n+1} - \left(\frac{\alpha\Delta t}{\Delta x^2}\right)\varepsilon_{m+1,n} - \left(1 - 2\frac{\alpha\Delta t}{\Delta x^2}\right)\varepsilon_{m,n} - \left(\frac{\alpha\Delta t}{\Delta x^2}\right)\varepsilon_{m-1,n} = R_{E_{m,n+1}} + T_{E_{m,n+1}} \tag{E.67}$$

这是一个二维非齐次有限差分方程，表示了（8.22）抛物型偏微分方程数值求解过程中的误差传播。这个有限差分方程的解很难求，因此，先用冯·诺依曼法来分析式（E.67）的齐次部分，即

$$\varepsilon_{m,n+1} - \left(\frac{\alpha\Delta t}{\Delta x^2}\right)\varepsilon_{m+1,n} - \left(1 - 2\frac{\alpha\Delta t}{\Delta x^2}\right)\varepsilon_{m,n} - \left(\frac{\alpha\Delta t}{\Delta x^2}\right)\varepsilon_{m-1,n} = 0 \tag{E.68}$$

这个方程只表示了在初始计算点（即 $n=0$ 处）引入的误差的传播，忽略了 $n>0$ 时产生的截断误差和舍入误差。

这个齐次有限差分方程的解可以写成如下分离的形式：

$$\varepsilon_{m,n} = c\mathrm{e}^{\gamma n\Delta t}\mathrm{e}^{i\beta m\Delta x} \tag{E.69}$$

其中，$i=\sqrt{-1}$，c、γ 和 β 为常数。当 $n=0$ 时，有

$$\varepsilon_{m,0} = c\mathrm{e}^{i\beta m\Delta x} \tag{E.70}$$

这是初始计算点的误差，因此，$\mathrm{e}^{\gamma\Delta t}$ 这一项可以看作是初始误差的放大因子。为了使原始误差不随 n 的增加而增大，这个放大因子必须满足冯·诺依曼稳定条

件，即

$$|e^{\gamma\Delta t}| \leqslant 1 \tag{E.71}$$

放大因子可以是复数，复数的模必须满足以上不等式，也就是

$$|r| \leqslant 1 \tag{E.72}$$

因此，如图 E.3 所示，复平面上的稳定区域是一个半径为 1 的圆。

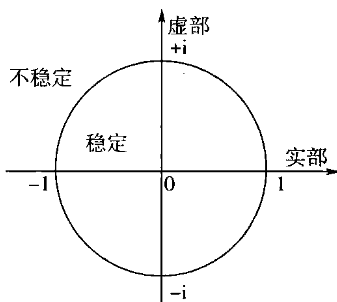


图 E.3　复平面上的稳定区域

将式（E.69）代入式（E.68），并重排，可以求得放大因子：

$$e^{\gamma\Delta t} = \left(1 - 2\frac{\alpha\Delta t}{\Delta x^2}\right) + \frac{\alpha\Delta t}{\Delta x^2}(e^{i\beta\Delta x} + e^{-i\beta\Delta x}) \tag{E.73}$$

利用三角恒等式

$$\frac{e^{i\beta\Delta x} + e^{-i\beta\Delta x}}{2} = \cos(\beta\Delta x) \tag{E.74}$$

和

$$1 - \cos(\beta\Delta x) = 2\sin^2\left(\frac{\beta\Delta x}{2}\right) \tag{E.75}$$

可以将式（E.73）化为

$$e^{\gamma\Delta t} = 1 - \left(4\frac{\alpha\Delta t}{\Delta x^2}\right)\left[\sin^2\left(\frac{\beta\Delta x}{2}\right)\right] \tag{E.76}$$

代入冯·诺依曼稳定条件，可得稳定界限为

$$0 \leqslant \left(\frac{\alpha\Delta t}{\Delta x^2}\right)\left[\sin^2\left(\frac{\beta\Delta x}{2}\right)\right] \leqslant \frac{1}{2} \tag{E.77}$$

其中 $\sin^2(\beta\Delta x/2)$ 的最大值为 1，因此

$$0 \leqslant \left(\frac{\alpha\Delta t}{\Delta x^2}\right) \leqslant \frac{1}{2} \tag{E.78}$$

这就是这种算法的条件稳定界限。应该注意，这个界限与 8.5.2 节应用正性法则得到的结果是一样的。

利用冯·诺依曼方法同样可以分析双曲型方程（8.82）的显式解（8.84）的稳定性。该解的误差传播齐次方程为

$$\varepsilon_{m,n+1}-2\left(1-\frac{\alpha^2\Delta t^2}{\Delta x^2}\right)\varepsilon_{m,n}-\frac{\alpha^2\Delta t^2}{\Delta x^2}(\varepsilon_{m+1,n}+\varepsilon_{m-1,n})+\varepsilon_{m,n-1}=0 \quad (E.79)$$

将式（E.69）代入式（E.79），并用三角恒等式（E.74）和式（E.75），得到放大因子：

$$e^{\gamma\Delta t}=\left[1-2\frac{\alpha^2\Delta t^2}{\Delta x^2}\sin^2\left(\frac{\beta\Delta x}{2}\right)\right]\pm\sqrt{\left[1-2\frac{\alpha^2\Delta t^2}{\Delta x^2}\sin^2\left(\frac{\beta\Delta x}{2}\right)\right]^2-1} \quad (E.80)$$

这个放大因子在复平面上要求满足不等式（E.72），也就是

$$\left[1-2\frac{\alpha^2\Delta t^2}{\Delta x^2}\sin^2\left(\frac{\beta\Delta x}{2}\right)\right]^2-1\leqslant 0 \quad (E.81)$$

此式可以转化成如下不等式

$$\frac{\alpha^2\Delta t^2}{\Delta x^2}\leqslant\frac{1}{\sin^2\left(\frac{\beta\Delta x}{2}\right)} \quad (E.82)$$

其中 $\sin^2(\beta\Delta x/2)$ 的最大值为 1，因此

$$\frac{\alpha^2\Delta t^2}{\Delta x^2}\leqslant 1 \quad (E.83)$$

这就是这种算法的条件稳定界限。

用同样的方法还可以考察其他显式和隐式有限差分算法的稳定性，Lapidus 和 Pinder（1982）两人已完成了这些工作，并得出结论：大多数显式有限差分近似算法是条件稳定的，而多数隐式近似算法是绝对稳定的。

E.5 步长控制

为了简化，前几节的稳定性分析都假设整个积分过程中 λ 值不变，对于式（E.1）这样的线性方程，这个假设成立；但对于非线性方程，在积分区间上，λ 值会有很大变化。这时就要用可能的最大 λ 值来确定最小的积分步长，这样就可以用计算时间换取稳定性。如果计算时间过长，就要采取策略使每一步积分能自动调节步长。

检验步长的一种简单测试方法就是每个区间计算 2 次，用全步长计算一次，再用较小的步长（通常取第一次步长的一半）计算一次。积分区间计算完成时，如果两种步长求得的 y 值之差小于给定的收敛标准，那么步长可以增加；否则，如果两个 y 值之差超出了允许的范围，就说明步长太大，需要缩短步长才能使截断误差控制在允许范围之内。

还有一个控制步长的方法就是在每个区间上估计截断误差。龙格-库塔-费尔贝格法就采用了这种方法（参见 Constantinides 和 Mostoufi（1999）著作的表 5.2），其中有局部截断误差的估算。在计算机程序中很容易加入误差估算功能，这样，可以让程序在每一步计算中自动改变步长，直至达到所要求的精度。

如前所述，最佳校正次数为 2，因此，如果采用预测-校正法，若在第二个校正值之前就收敛了，则可以加大步长；另一方面，若使用第二个校正值之后仍不能收敛，则需要减小步长。

E.6　刚性微分方程组

在 E.1 节中，我们已经看到微分方程数值解的稳定性取决于 $h\lambda$ 的值，并且 λ 与算法的稳定界限一起决定了积分步长。对于线性微分方程

$$\frac{\mathrm{d}y}{\mathrm{d}t} = \lambda y \tag{E.84}$$

λ 为方程的特征值，并且在整个积分过程中其值保持不变，为常数。对于非线性微分方程

$$\frac{\mathrm{d}y}{\mathrm{d}t} = f(t, y) \tag{E.85}$$

可以应用中值定理（E.26）将每一步计算线性化，这样，由函数对 y 的偏导数可以求得 λ 值：

$$\lambda = \left.\frac{\partial f}{\partial y}\right|_{\alpha, t_n} \tag{E.86}$$

这里的 λ 不再是常数，其大小在每一步积分中都会变化。

这种方法可以推广到如下非线性微分方程组：

$$\begin{aligned} \frac{\mathrm{d}y_1}{\mathrm{d}t} &= f_1(t, y_1, y_2, \cdots, y_n) \\ \frac{\mathrm{d}y_2}{\mathrm{d}t} &= f_2(t, y_1, y_2, \cdots, y_n) \\ &\vdots \\ \frac{\mathrm{d}y_n}{\mathrm{d}t} &= f_n(t, y_1, y_2, \cdots, y_n) \end{aligned} \tag{E.87}$$

将方程组线性化，产生雅可比矩阵

$$J = \begin{bmatrix} \frac{\partial f_1}{\partial y_1} & \cdots & \frac{\partial f_1}{\partial y_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial y_1} & \cdots & \frac{\partial f_n}{\partial y_n} \end{bmatrix} \tag{E.88}$$

雅可比矩阵的特征值 $\{\lambda_i | i = 1,2,\cdots,n\}$ 是数值解稳定性分析的决定因素。积分步长则取决于算法的稳定界限和最大特征值。

如果微分方程雅可比矩阵特征值的大小都处于同一个数量级，那么方程组求积分不会有什么问题。但是，如果特征值中最大值比最小值要大好几个数量级，那么方程组被称为刚性的。这种方程组的刚性比（Stiffness Ratio，SR）定义为

$$SR = \frac{\max\limits_{1 \leqslant i \leqslant n} |\mathrm{Real}(\lambda_i)|}{\min\limits_{1 \leqslant i \leqslant n} |\mathrm{Real}(\lambda_i)|} \tag{E.89}$$

积分步长由最大特征值决定，而积分时间通常由最小特征值决定，这是因为受最小特征值影响最大的那些变量，其变化很慢，因此，用显式法积分刚性微分方程组，会花费很多计算机时间。

MATLAB 的函数 ode23s 和 ode15s 就是适用于刚性常微分方程求解的解法器(参见表 7.2)。

E.7 参考文献

Constantinides, A. and Mostoufi, N. 1999. *Numerical Methods for Chemical Engineers with MATLAB Applications*. Upper Saddle River, NJ: Prentice Hall PTR.

Lapidus, L. and Sienfeld J. H. 1971. *Numerical Solution of Ordinary Differential Equations*. New York: Academic Press.

Lapidus, L. and Pinder G. F. 1982. *Numerical Solution of Partial Differential Equations in Science and Engineering*, New York: J. Wiley & Sons, Inc.

国际视野　科技前沿

国际信息工程先进技术译丛

《WCDWA原理与开发设计》

《下一代移动系统：3G/B3G》

《IMS:IP多媒体概念和服务》（原书第2版）

《下一代无线系统与网络》

《深入浅出UMTS无线网络建模、规划与自动优化：理论与实践》

《HSDPA/HSUPA技术与系统设计——第三代移动通信系统宽带无线接入》

《无线传感器及元器件：网络、设计与应用》

《印制电路板——设计、制造、装配与测试》

《IPTV与网络视频：拓展广播电视的应用范围》

《多电压CMOS电路设计》

《微电子技术原理、设计与应用》

《蜂窝网络高级规划与优化2G/2.5G/3G/…向4G的演进》

《基于蜂窝系统的IMS——融合电信领域的VoIP演进》

《无线网络中的合作原理与应用》

《电生理学方法与仪器入门》

《移动电视：DVB-H、DMB、3G系统和富媒体应用》

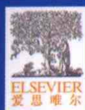
《环境网络：支持下一代无线业务的多域协同网络》

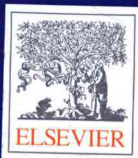
《基于射频工程的UMTS空中接口设计与网络运行》

《未来UMTS的系统结构与业务平台：全IP的3G CDMA网络》

《UMTS-HSDPA系统的TCP性能》

《数值方法在生物医学工程中的应用》





上架指导：工业技术/电气工程



ISBN 978-7-111-25365-5

封面设计：马精明

编辑热线：(010)88379178

地　址：北京市百万庄大街22号　邮政编码：100037
联系电话：(010)68326294　(010)68993821
网址：http://www.cmpbook.com（机工门户网）
E-mail:cmp@cmpbook.com
购书热线：(010)88379639　(010)88379641　(010)88379643

定价：78.00元